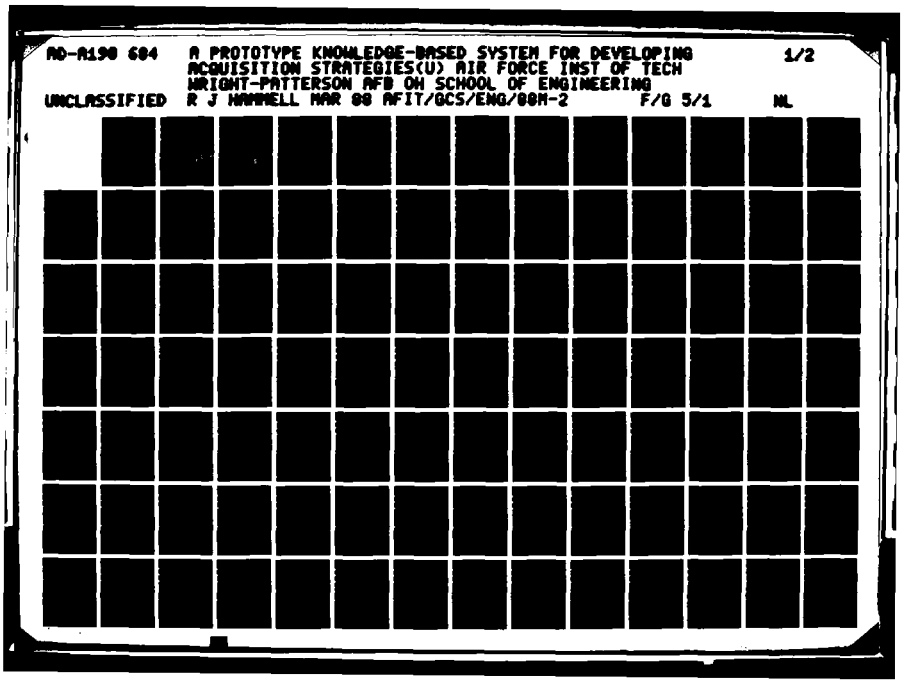
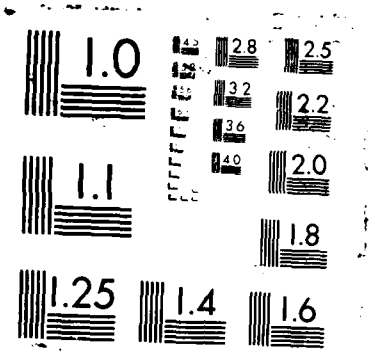

AD-A190 604
A PROTOTYPE KNOWLEDGE-BASED SYSTEM FOR DEVELOPING ACQUISITION STRATEGIES(U) AIR FORCE INST OF TECH WRIGHT-PATTERSON AFB OH SCHOOL OF ENGINEERING
1/2
UNCLASSIFIED
R J HAMMELL MAR 88 AFIT/GCS/ENG/88M-2
F/G 5/1
NL


1.0
1.1
1.25
1.4
1.6
1.8
2.0
2.2
2.5
2.8
3.2
3.6
4.0

A PROTOTYPE KNOWLEDGE-BASED SYSTEM FOR DEVELOPING ACQUISITION STRATEGIES

THESIS

Robert J. Hammell, II
Captain, USA

AFIT/GCS/ENG/88M-2

DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY
AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

AFIT/GCS/ENG/88

A PROTOTYPE KNOWLEDGE-BASED SYSTEM FOR DEVELOPING ACQUISITION STRATEGIES

THESIS

Robert J. Hammell, II
Captain, USA

AFIT/GCS/ENG/88M-2

AFIT/GCS/ENG/88M-2

A PROTOTYPE KNOWLEDGE-BASED SYSTEM FOR DEVELOPING ACQUISITION STRATEGIES

THESIS

Presented to the Faculty of the School of Engineering

of the Air Force Institute of Technology

Air University

In Partial Fulfillment of the

Requirements for the Degree of

Master of Science in Computer Systems

Robert J. Hammell, II, B.S.

Captain, USA

March 1988

Preface

The purpose of this thesis was to demonstrate the effectiveness of applying knowledge-based system techniques to an actual Army problem. This research addressed the general area of program management, and specifically the problem of developing acquisition strategies for an Army procurement. The prototype system developed in this thesis clearly shows that knowledge-based systems can greatly assist in the generation of acquisition strategies, and that tremendous potential exists for providing "intelligent assistants" to help generate textual documents.

The success of this project is obviously the result of contributions from many people. I could not have undertaken this effort alone and I am deeply grateful for the friendship and assistance I received throughout the research.

I must first thank Dan Causey, Tammie Vogler, and Brian David at the Belvoir Research, Development and Engineering Center (BELVOIR). These individuals served as my "experts" and provided the knowledge for the prototype system developed in this research. They endured many questions and this project could not have even been started, much less completed, without their assistance. Although too numerous to mention individually, I would like to thank the many other people within BELVOIR that took time out of their busy schedules to offer key input, expertise, and advice.

Thanks is also due to Mr. Hal Schutt of the Defense Systems Management College for sponsoring this thesis. He made numerous trips from Fort Belvoir, Virginia and was always willing to provide the necessary assistance to make this thesis a quality product.

I am deeply indebted to my thesis advisor, Major Steve Cross, for his many suggestions and guidance. He managed to help me maintain forward progress while still allowing me to perform independent research; a delicate balance to maintain. Although extremely busy with a new job, he still found time to provide critical insight and ideas to help me through the "rough spots." The workplace, software, computer equipment, and funding he secured for me made this thesis possible.

Thanks must also go to the other members of my thesis committee: Lieutenant Colonel Gregory Parnell, Major Steven LeClair, and Captain Wade Shaw. They did so much more than just act as readers, and their ideas and suggestions are reflected throughout this thesis. Thanks is also due to Captain Tom Triscari, who served in the same capacity as a committee member and provided many good ideas.

I would like to thank Lieutenant Steve Wagner, who was not only a good friend but also a great sounding board for ideas. His sense of humor allowed me to keep things in perspective. Tacos on me, Steve.

My deepest thanks go to my wife, Michelle. Her love, encouragement, and constant support were instrumental in my ability to complete this thesis, as well as my graduate education. She not only endured many long nights while I worked on this thesis, but endured many long nights of her own in typing it. As always, she was selfless in her assistance and words cannot do justice to the gratitude I feel. To her I dedicate this thesis. Thanks, Chelle.

Robert J. Hammell, II

Table of Contents

List of Figures

List of Tables

Abstract

The acquisition strategy is one of the most important program management documents. The diverse nature of the information required in an acquisition strategy, coupled with frequent changes in acquisition policy and normal loss of organizational expertise in a military environment, create significant problems for the developer of an acquisition strategy.

This thesis demonstrates the effectiveness of applying knowledge-based system techniques to the problem of developing acquisition strategies for government procurements. The need for automated assistance for generating an acquisition strategy is examined, and the reasons why conventional programming techniques are inadequate for this problem are discussed. The development of a prototype knowledge-based system using Guru is outlined, and an evaluation of the system is presented.

The prototype system addresses Non-Developmental Item (NDI) procurements, and two of the thirteen required acquisition strategy paragraphs are implemented. A capability to edit the paragraphs is provided, and consistency between the paragraphs is forced during paragraph development as well as during paragraph editing.

This thesis concluded that a knowledge-based "intelligent assistant" could provide significant benefit to developers of an acquisition strategy. Possible enhancements to extend the prototype and potential areas of further research are also discussed.

A PROTOTYPE KNOWLEDGE-BASED SYSTEM

FOR DEVELOPING ACQUISITION STRATEGIES

I. INTRODUCTION

Background

The Defense Systems Management College (DSMC), located at Fort Belvoir, Virginia, is a Department of Defense (DOD) organization with three primary defense acquisition missions. The first mission is one of education. The DSMC provides education in program management policies, philosophies, skills, and techniques to train members of the defense acquisition community, primarily program management office (PMO) personnel. The second mission is that of research. The DSMC also conducts research in applied management science to complement the education mission and to support DOD acquisition efforts. The third mission of the DSMC is to disseminate information throughout the DOD acquisition community. (Defense Systems Management College, 1987:1)

As part of the execution of the above missions, the Program Manager's Support System Directorate was created in August 1983 (Defense Systems Management College, 1987:pg 3). The directorate is currently headed by Mr. Harold Schutt. The primary mission of this directorate is the development of the Program Manager's Support System (PMSS), an integrated software system aimed at assisting defense system program managers in the management of their programs. The PMSS is an application of decision support systems technology to the defense acquisition program management environment. The PMSS consists of two major parts: the

functional modules and the integrated PMSS. The functional modules are software programs that can be used in a stand-alone mode to support specific program management functions. The integrated PMSS combines the functional modules to provide a Program Overview capability. Many functional modules are already completed, with others in various stages of development. The integrated PMSS is scheduled for completion in fiscal year 1988. (Defense Systems Management College, 1987:i-ii, 24-26)

As an addition to the PMSS, the DSMC is preparing to contract for the development of a Procurement Package Generator module. When completed, this module will be used to assist in the generation of Commerce Business Daily announcements, Statements of Work, Specifications, Contract Data Requirement Lists, Acquisition Strategies, and other procurement documents. Various software programs exist throughout the Department of Defense that already contain most of the needed capabilities, and the effort will largely be one of standardization and integration. (Defense Systems Management College, 1987:69)

Of the documents to be included in the Procurement Package Generator module, the acquisition strategy is one of the most important. An acquisition strategy is required for all materiel acquisition programs regardless of dollar amount. The primary purpose of the acquisition strategy is to document the general concepts that serve as direction for the program throughout the entire acquisition cycle. (AMC/TRADOC PAM 70-2, 1987:7.1) Thus, it is the program's guide for the development, support, production, and deployment of equipment. The acquisition strategy is structured in a prescribed format and must not exceed 15 pages. Thirteen specific elements must be addressed in the acquisition strategy and if any of the elements do not apply to a

particular program, the rationale must be provided (AMC/TRADOC PAM 70-2, 1987:7.10). It is a "living document;" that is, the acquisition strategy is updated throughout the acquisition life cycle. The information contained in the acquisition strategy is typically general in the early phases of a program with the level of detail increasing as the program matures (AMC/TRADOC PAM 70-2, 1987:7.1).

In addition to being one of the most important program documents, the acquisition strategy is one of the most difficult to develop. The elements that must be addressed in the document are very diverse. Although the person responsible for developing the acquisition strategy usually has detailed knowledge about the program itself, he seldom knows enough about the variety of topics that must be addressed to do it alone. As just one example, formulation of the detailed data required for paragraph 2, Contracting Strategy, usually requires the assistance of someone knowledgeable and experienced in the contracting area. Many of the other topics that must be addressed also require assistance from, and coordination with, personnel having other specific domains of expertise. The excessive time required to coordinate with these individuals (who can be in different buildings, different organizations, and so on) slows the development process of the document considerably, as well as the staffing process. In addition, the personnel whose assistance is needed usually have many ongoing programs and projects to support at any given time, which makes their assistance even more difficult to schedule.

The development of the paragraphs for which outside help is not needed is still difficult. This is due to the frequent rate at which acquisition policy, guidance, and procedures change. Also, it is

typically the case that the project engineer (also called the program manager or project manager, depending on the organization) has never gone through the acquisition strategy development process before and has no personal knowledge or experience to draw on. The frequent changes in policy and the lack of personal experience, coupled with the normal high turnover rate in the military environment which causes considerable loss of organizational experience and expertise, creates a steep learning curve for the preparation of an acquisition strategy.

Other problems also exist with developing and updating the acquisition strategy. In some situations, an approved acquisition strategy for a similar procurement can be found to use as an example. However, the inability to capture the rationale for why the document was developed as it was, and why certain changes may have been made, limits the usefulness of the examples. Since input from many people is required for an acquisition strategy, and since the same people (even within one organization) often do not work on all acquisition strategies, inconsistencies can (and do) exist between acquisition strategies even if they are for similar procurements.

Problem

Although the acquisition strategy is an important document and difficult to develop, no software programs currently exist that provide assistance for the developer of an acquisition strategy. For the most part, this is due to the difficulty of handling the acquisition strategy development problems with "conventional" computer programming techniques. Specifically, conventional computer programming techniques do not allow for the efficient representation and manipulation of

"knowledge" (ie. the experience of personnel who have developed acquisition strategies and/or the expertise of personnel in a specialized area such as contracting). Additionally, computer programs are traditionally not well suited to frequent changes (such as the frequent changes in acquisition policies and procedures).

The emergence of knowledge-based systems, however, has provided new techniques that have proved beneficial in solving problems analogous to those in the acquisition strategy development area. (A detailed discussion of knowledge-based systems, including their definition, a comparison with conventional programs, and guidelines for their use, is presented in Chapter II.) Before the DSMC can fully develop the Procurement Package Generator module, the capability for automated assistance for developing acquisition strategies must be available. Therefore, the DSMC has structured development of the Procurement Package Generator module in two phases, with Phase I being the creation of a knowledge-based system to generate acquisition strategies.

The primary purpose of this research was to develop a prototype knowledge-based system to demonstrate the effectiveness of applying knowledge-based system techniques to the problem of developing an acquisition strategy. The prototype system developed in the course of this research will not only be used to provide a basis for contracting the development of a "full blown" knowledge-based system to generate acquisition strategies, but will also be used to provide insight to the DSMC (and other organizations that will be involved) about what to expect from a knowledge-based system, the level of effort required to develop one, and the types of people ("experts") that must be made available during the development process. By using the prototype for

demonstration purposes, the expectations of the government "experts" that will be involved in the development of the complete system can be bounded from the start, and a basis for the effort they will be asked to provide can be established. In addition, the prototype system will hopefully furnish the selected contractor with a foundation on which to expand by providing him at least a small knowledge base as a start, as well as a framework of features and techniques. Feedback from demonstrations of the prototype to the target users should prove invaluable by allowing government personnel to observe the capabilities of a knowledge-based system, and by stimulating suggestions for additions or modifications to the features contained in the prototype.

A detailed problem assessment is presented in Chapter III.

Scope

To validate the concept of using a knowledge-based system for the problem of generating acquisition strategies, the capability of the prototype knowledge-based system was limited to the generation of initial draft acquisition strategies for Non-Developmental Item (NDI) procurements by the Belvoir Research, Development and Engineering Center (BELVOIR). All procurements by BELVOIR are in the $10,000 to $1,000,000 range, which are classified as non-major weapon systems, and the NDI approach is applicable to many of the BELVOIR programs. To further define the scope of the prototype knowledge-based system, a limited number of representative acquisition strategies for BELVOIR was examined. The design, capability, and evaluation of the system was based on these representative cases.

Since the amount of knowledge required to be able to develop an

acquisition strategy is so large, time constraints on both the researcher and available experts precluded the ability to address all thirteen acquisition strategy paragraphs. To validate the concept of using the knowledge-based system approach, paragraphs 1 and 3 (Program Structure and Tailoring the Acquisition Process, respectively) were selected for implementation. The basis for this selection is presented in Chapter III.

DSMC and BELVOIR personnel both agreed that the scope outlined above was sufficient to demonstrate the applicability of the knowledge-based system approach. Paragraphs 1 and 3 were rated as two of the four most difficult acquisition strategy paragraphs to develop, and it was agreed that the implementation of these two in the prototype would demonstrate the ability of a knowledge-based system to handle the problem of assisting with the generation of the entire document. The limitation of addressing a small number of representative NDI procurements by BELVOIR was done strictly to scope the problem to fall within time constraints imposed on this research. It was agreed that this limitation was reasonable, and that the prototype would still provide more than adequate evidence that knowledge-based systems could solve the more difficult (ie., more knowledge intensive) problem of assisting with acquisition strategies for procurement of all types.

Standards

The goal of any knowledge-based system is to produce an "acceptable answer." An acceptable answer is defined as one with which the expert(s) would agree, both in terms of the final answer and the reasoning process used to reach that answer.

The prototype knowledge-based system developed in this research was evaluated using the following criteria:

- System performance;
  - -- "Correctness" of knowledge,
  - -- "Correctness" of lines of reasoning,
  - -- "Correctness" and "completeness" of written text,
  - -- User interface,
  - -- Verification of code,
- Maintainability;
  - -- Ease (or difficulty) in correcting "bad" knowledge, lines of reasoning, and/or written text,
- Expansion;
  - -- Ease (or difficulty) in adding knowledge and capabilities to implemented paragraphs and in implementing others within the established system framework, and
- Applicability of the knowledge-based system approach to the chosen problem domain.

Approach

The following approach was used to meet the objectives of this research:

1. Investigated procedures for knowledge acquisition.
2. Completed a detailed assessment of the problem to be solved by the prototype system.
3. Developed a knowledge acquisition approach and acquired knowledge from the appropriate expert(s).
4. Designed a prototype knowledge-based system to generate acquisition strategies for the Belvoir Research, Development, and Engineering Center within the previously defined scope.
5. Evaluated available knowledge-based system development tools and selected one appropriate for the problem domain.

6. Built and tested the prototype system.

7. Designed a methodology for evaluation of the prototype system.

8. Evaluated the prototype system and the applicability of the knowledge-based system approach to the particular problem domain.

The remainder of this thesis is organized as follows: Chapter II presents a review of current literature applicable to this research. In Chapter III, the detailed problem assessment is provided. Chapter IV addresses the design and implementation of the prototype and Chapter V provides the prototype evaluation. In Chapter VI, conclusions and recommendations resulting from this research are presented.

II. LITERATURE REVIEW

In this chapter, a review of current literature pertinent to this research effort is presented. First, knowledge-based systems are defined. Then, knowledge-based systems are compared to conventional programs, guidelines for when to use knowledge-based systems are presented, and the process of knowledge acquisition is discussed. Finally, research related to this research effort is reviewed.

Knowledge-Based Systems

Since the core of this research effort deals with the development of a knowledge-based system, it is both appropriate and necessary to first establish what "knowledge-based system" means. This is especially true considering the confusion that can result when the term "expert system" is used interchangeably with "knowledge-based system" by some, but not by all. The objective of this section is not to try and define these terms once and for all, but rather to establish their meanings in relation to this research effort.

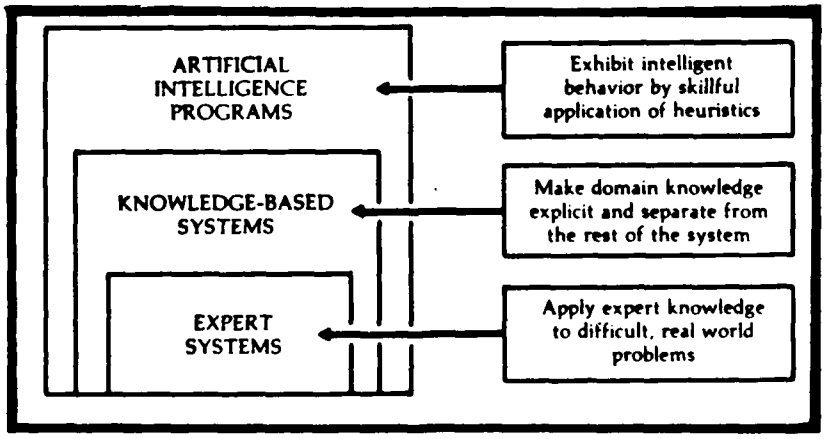
Knowledge-based systems and expert systems have many similarities. First, they are both computer programs. They both also contain representations of (human) knowledge, and use an inference procedure to manipulate the knowledge. Both systems are designed to use their knowledge and inference procedures to solve problems in some particular (usually narrow) problem domain. (Rosenberg, 1986:101)

The key to understanding what little difference exists between the two lies in their design and expected level of performance. Figure 1 and the following from Waterman reinforce the concept that knowledge-based systems and expert systems overlap, and hints at the idea that the

difference is in the level of "true human expert" modeling that takes place:

> The knowledge in an expert system is organized in a way that separates the knowledge about the problem domain from the system's other knowledge, such as general knowledg about how to solve problems or knowledge about how to interact with the user--for example, how to print characters at the user's terminal or modify lines of data in response to the user's commands. This collection of domain knowledge is called the knowledge base, while the general problem-solving knowledge is called the inference engine. A program with knowledge organized in this way is called a knowledge-based system.
>
> ... virtually all expert systems are knowledge-based systems, while the converse is not necessarily true.
>
> (Waterman, 1986:18)

> Expert system: A computer program using exper nowledge to attain high levels of performance in a narrow p: m area.
>
> (Wate 1986:11)



(Waterman, 1986:18)

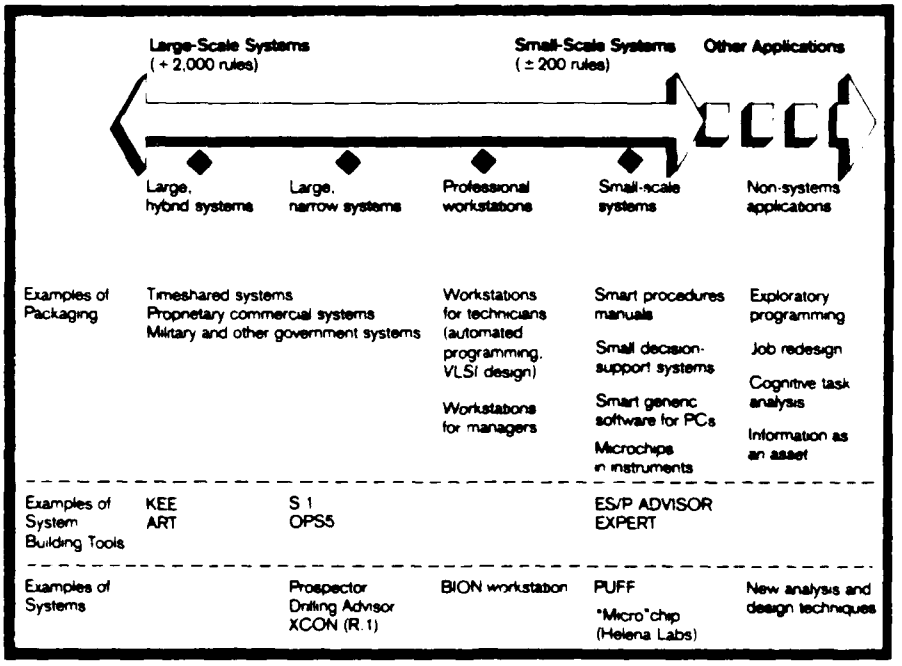
Figure 1.

Overlap Between Expert Systems and Knowledge-Based Systems

While Waterman suggests that expert systems are a category of knowledge-based systems, Harmon and King present a slightly different

view. Figure 2 illustrates the range of applications for expert systems and defines "Small-Scale Systems" as follows:

> Such systems can be run on personal computers and might be called intelligent decision aids. They are knowledge-based systems designed to help individuals deal with small but nasty problems. In some cases, these problems require knowledge engineering techniques because they are not easily amenable to traditional solutions. In other cases, the problems are simply ones that users understand and want to solve. Users may not understand the traditional programming techniques that would be required to solve these problems, but they would be able to turn to knowledge-based approaches simply because of the user-friendly interface and the rapid prototyping features that are available.
>
> (Harmon and King, 1985:232)



(Harmon and King, 1985:228)

Figure 2

Expert System Applications

Although the view presented by Harmon and King is slightly different than that presented by Waterman, combining the two supplies sufficient clarity for definition. Both expert systems and knowledge-based systems contain a "knowledge base," and an "inference engine" to use the knowledge in problem solving. Expert systems are designed to model the reasoning process of a human expert and are expected to perform at or near the level of a human expert. Knowledge-based systems are not necessarily designed to model the human expert exactly; they are designed simply to apply knowledge (at least some of it from a human "expert") and inference procedures to solve problems (maybe smaller, less general problems than expected of an expert system). While this implies that some degree of "human" reasoning will be inherent in the knowledge-based system, performance is expected to be more in the direction of an "intelligent assistant" than at the level of a "true human expert." It is convenient to think of expert systems as special cases of knowledge-based systems.

Due to arguments and criticisms that could result from semantic interpretations of "human expert," "performance at or near the level of a human expert" and other terms, this research claims to operate under the more general definition of knowledge-based systems. Depending upon the reader's point of view, the prototype knowledge-based system developed in this research could be considered an expert system.

The most important point is to realize that a knowledge-based system can provide significant benefits regardless of whether or not performance at the "expert" level is attained. At the most basic level, a knowledge-based system can apply knowledge from several individuals, as well as from manuals and written policy, to assist users in their

jobs. In the case of developing an acquisition strategy this is particularly beneficial due to the wide range of topics that must be addressed. A knowledge-based system containing knowledge from contracting specialists, logistic specialists, and other specialists would provide a project engineer (who develops the acquisition strategy) with access to much more expertise and experience than he usually has on his own. If "knowledge" from regulatory guidance and local policy is also included, it is easy to see that significant increases in productivity and quality would be possible through the use of a knowledge-based system.

To summarize, within the scope of this research a knowledge-based system is considered to be a computer program that uses knowledge and inference procedures to solve problems. A knowledge-based system can be considered to be an expert system with the constraints for exactly modeling the reasoning process of a "true human expert" and for performance at the level of a "true human expert" removed. It is for this reason that, throughout the research, all techniques and guidelines in the literature that apply to expert system development are considered to apply to the development of knowledge-based systems as well. As such, much of the information referenced throughout the remainder of this chapter was originally written to address issues related to expert systems but is presented here to address issues related to the more general category of knowledge-based systems.

Knowledge-Based Systems Versus Conventional Programs

To assist in understanding when knowledge-based methodologies should (and should not) be used, it is important to briefly compare knowledge-based systems with conventional programs. According to

(Waterman, 1986:24), Teknowledge (a company specializing in the development of knowledge-based systems) characterizes the differences between knowledge-based systems and conventional computer programs as follows:

Conventional Programs	Knowledge-Based Systems
Representation and use of data	Representation and use of knowledge
Algorithmic	Heuristic
Repetitive process	Inferential process
Effective manipulation of large data bases	Effective manipulation of large knowledge bases

As previously noted, a key characteristic of a knowledge-based system is the knowledge base. In fact, the Expert Systems Handbook produced by the National Aeronautics and Space Administration (NASA) states "...it is the emphasis on human knowledge about how to solve specific problems" that is the main separator of knowledge-based systems from other computer programs (NASA, 1986: 1-1).

Another key aspect of a knowledge-based system is the inferential process, or the ability to "reason" over the knowledge it contains. For example, suppose the system "knows" that a Non-Developmental Item (NDI) Category A procurement is always done with a Firm-Fixed Price Contract, and that Program X is NDI Category A. It should then be able to "reason" over these two facts and answer the question "What type of contract will be used for Program X ?" (Barr and others, 1981:Ch 3).

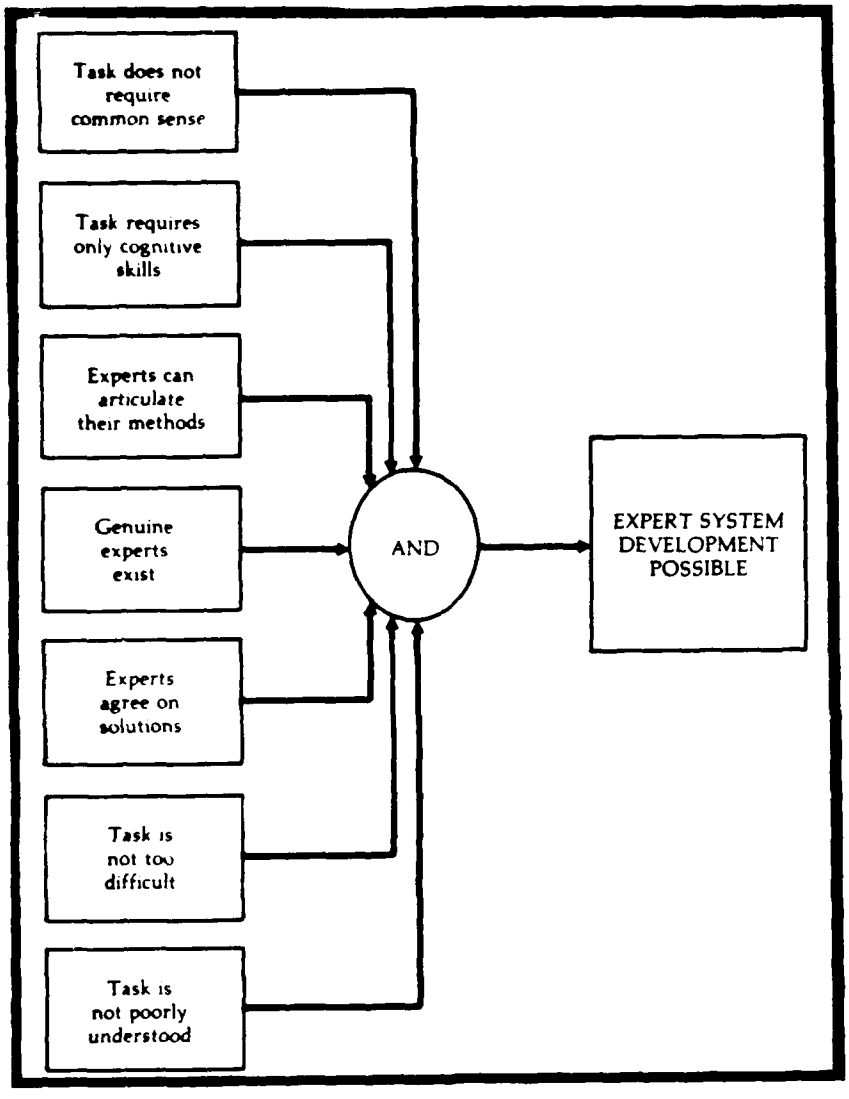
Finally, another important difference between knowledge-based systems and conventional computer programs is the problem solving techniques used in each. Conventional computer programs are algorithmic in

nature; that is, they are designed around formal procedures (algorithms) guaranteed to produce correct answers every time (Waterman, 1986:22,29). Knowledge-based systems, however, are designed to solve problems by reasoning; that is, using strategies and heuristics (rules-of-thumb) to solve problems. The most important point here is that knowledge-based systems, by design, will behave more like a human: they will usually produce correct or acceptable results but will occasionally make mistakes (Waterman, 1986:29). The fact that knowledge-based systems are not even designed to be infallible is a key factor in understanding when they should be considered for use.

When Should a Knowledge-Based System Be Used?

The development of a knowledge-based system is a complex, time-consuming, costly task (Bobrow, 1986:880-893; Waterman, 1986:Ch 11; Prerau, 1985:26-30). Just as the design of a knowledge-based system and the knowledge it should contain is very problem-dependent, so is the decision of whether or not to use a knowledge-based system. There is no list of suitable problems; rather, a problem must be examined to see if it lends itself to a solution by the knowledge-based system approach.

Waterman offers general guidelines for when to use knowledge-based systems to solve a problem. Figure 3 illustrates the requirements that a problem should meet to make knowledge-based system development possible. The task should not require common sense to be solved, since common sense is a broad spectrum of general knowledge that is currently not possible to efficiently represent in a computer. The problem should only require cognitive (thinking) skills to be solved, as opposed to physical skills that can only be learned from practice. The problem



(Waterman, 1986:129)

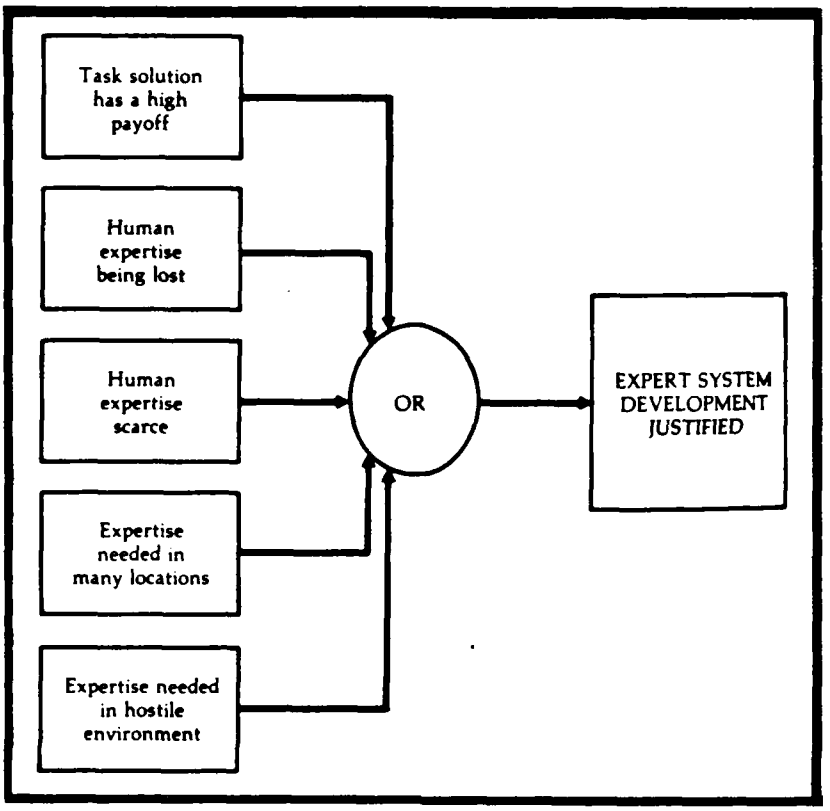
Figure 3

Requirements for Knowledge-Based System Development

must be understood well enough that the problem solving methods are known, and the task must not be so difficult that large bodies of different types of knowledge is needed (remember, the knowledge-based system is a "specialist"). Most importantly, a person with a high level of expertise at *solving the problem (the expert) must exist.* In addition, the expert must be able (and willing) to communicate the methods used to solve the problem, and there must be agreement among experts as to what constitutes a "good" solution (Waterman, 1986:128-129).

To justify knowledge-based system development, one or more of the conditions shown in Figure 4 should be met. Since knowledge-based system development is costly, a high payoff in terms of monetary savings or increased productivity is usually necessary to justify development. However, if human expertise is limited (thus expensive), is needed at many physically separated locations at once, or is constantly being lost through personnel changes (such as reassignment in the military), development of a knowledge-based system may also be justified. Finally, if expert decision making is needed in environments such as nuclear power plants and outer space where it is usually too expensive and/or dangerous to try and locate human experts, knowledge-based system development may well be justified (Waterman:130-131).

Knowledge-based system development is considered appropriate when the problem meets all the conditions outlined in Figure 5. As mentioned earlier, knowledge-based systems use strategies and rules-of-thumb (heuristics) to reason over large amounts of symbolically represented knowledge. Therefore, the nature of the problem must lend itself to this kind of representation. A suggested way to determine if a problem meets this criteria is called the telephone test: if the problem can be

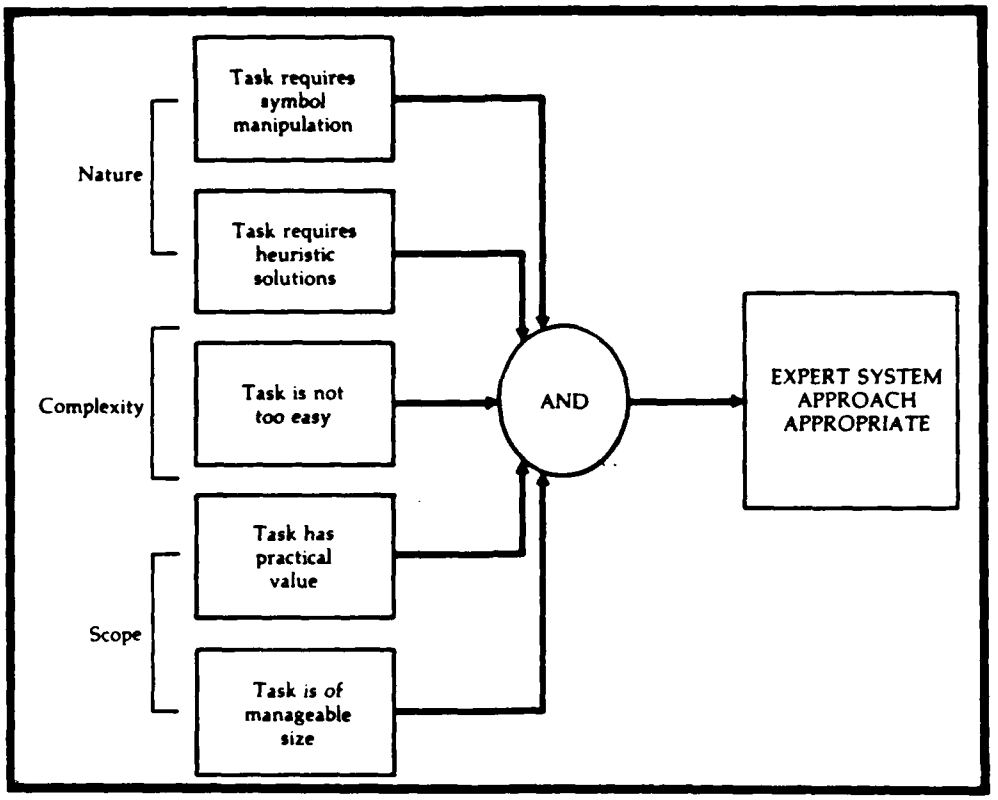


(Waterman, 1986:130)

Figure 4

Justification for Knowledge-Based System Development

explained over the telephone to an expert, and the expert can understand it and provide an answer, and the reasoning process can also be explained by the expert over the telephone, then the nature of the problem is probably suited to solution by a knowledge-based system. The problem should also be of sufficient difficulty to justify the time to develop the system. If a novice can be trained to solve the problem in a relatively short period of time, as opposed to a problem domain where it



(Waterman, 1986:32)

Figure 5

Characteristics that Make
Knowledge-Based System Use Appropriate

takes years of study and experience to learn how to solve the problem, it is not of sufficient complexity or difficulty to warrant the development of a knowledge-based system. Finally, the task should be a practical problem and be of manageable size. "Practical" and "manageable" are vague terms at best, and whether the problem is of "proper" scope is highly problem-dependent, but must be considered (Waterman, 1986:131-134).

In addition to determining whether knowledge-based system development is possible, justified, and appropriate, there are other

considerations that enter into the decision of whether or not to use the knowledge-based system approach. One key factor to consider is whether management is willing to support the development with monetary and personnel resources (Bobrow and others, 1986:886). Particularly important is the willingness to support the project with someone knowledgeable in the problem area. The "best" person at solving the problem is needed to provide the knowledge for the system, and management must be willing to free this person from normal responsibilities in support of system development.

Along the same lines, management must not have unrealistic expectations about how the knowledge-based system will perform. As stated earlier, a knowledge-based system will, to some degree, use the reasoning process of a human. Because of this, it will also sometimes make mistakes (Waterman, 1986:29). It must be enough that the knowledge-based system will perform at the level of a limited version of the human problem solver expert (Prerau, 1985:28). Some percentage of errors must be able to be tolerated. Also, it must be understood that the system will not be an "instant" expert (Bobrow and others, 1986:882). It will be an incremental process for the system to be given new knowledge, and for restructuring of old knowledge. John McDermott summarized the problems in this particular area very well in an article addressing the implementation of R1, a knowledge-based system used by Digital Equipment Corporation to configure orders for VAX-11 computer systems:

> Though R1 now has a large number of strong supporters at Digital and the extreme caution toward knowledge-based programs is waning, I have one remaining, quite general concern. It is not clear that all (or even most) of R1's supporters realize that R1 will always make mistakes. The problem is that at least some of R1's supporters think of it as a program rather than as an expert. There is, of course, a

> big difference between programs and experts. Finished programs, by definition, have no bugs. When experts are finished, on the other hand, they're dead. During the last two years, I have hammered on the theme that a knowledge-based program must pass through a relatively lengthy apprenticeship stage and then even after it has become an expert, it will, like all experts, occasionally make mistakes. The first part of this message got through, but I suspect that the second has not. My concern, then, is whether, as this characteristic of expert programs is recognized, Digital (or any large corporation) will be emotionally prepared to give a significant amount of responsibility to programs that are known to be fallible.
>
> (McDermott, 1981:26-30)

Even if management has the proper perspective about knowledge-based system development and performance, the end-user must also welcome the development of the system (Prerau, 1985:26-30). Many times the users feel that the purpose of the system is to replace them, instead of to give them assistance. If the user does not accept the system after it is developed, it may never be used. Feedback from the user should be considered before the system is designed. For example, suppose management of some airline decided that a percentage of errors (crashes) was tolerable and had a system built to fly airplanes. It might be an understatement to say that the system would probably not be met with enthusiastic user acceptance.

Prior to considering the knowledge-based system approach to solving the problem, conventional solutions should be considered (Prerau, 1985:26). Since there is usually less complexity, less technical risk, and less cost in using conventional programming approaches, it is better to use them if they are acceptable. This criteria would be well suited for inclusion in the conditions under which development of a knowledge-based system is justified.

Another point for consideration before developing a knowledge-based

system is the availability of test cases showing how the problem is normally solved by the expert (Bobrow, 1986:886; Prerau, 1986:29). Test cases not only provide a means by which to extract knowledge and reasoning strategies for the system, but also provide the means by which to evaluate the system's performance. The availability of test cases should be established before system development is begun since the test cases can provide insight as to whether the proposed system is feasible (Bobrow, 1986:886). If the problem is rare, relatively few test cases will be available. Also, if each solution method is unique, this will be illustrated by the sample test cases. In either event, the knowledge-based system approach to the problem would probably not be appropriate.

Finally, the issue of the existence of an expert and the ability of the expert to articulate the knowledge used to solve the problem is worth mentioning again. A good indicator as to whether the knowledge-based system approach is appropriate for solving a problem is the existence of one or more individuals that deal with the problem on a routine basis (Bobrow, 1986:). is not only enough for such a person to exist, but he must be able to communicate the knowledge, judgment, and reasoning methods used to solve the problem (Waterman, 1986:128; Prerau, 1985:28). This problem is best conveyed by the following quote:

> Unfortunately, the types of knowledge which are least easily elicited tend to be those which have built up over years of experience and hence form a vital part of an expert's expertise.
>
> (Berry and Broadbent, 1986:230)

Knowledge Acquisition

Because knowledge is an essential part of any knowledge-based system, it could (justifiably) be argued that the process of acquiring knowledge is the most important part in the development of any knowledge-based system (Prerau, 1987:43). Therefore, it is critical to understand what the knowledge acquisition process entails. This section will define what is meant by knowledge acquisition and present a discussion from current literature regarding methods of, and considerations during, knowledge acquisition.

Prerau states that the knowledge acquisition process is used to "find the knowledge that domain experts use to perform the task of interest" (Prerau, 1987:43). Knowledge acquisition not only includes finding the knowledge experts use, but also refining it within the context of solving problems in a particular domain. The process can include interviewing experts and/or observing experts while they solve problems. Although research of applicable manuals and literature can be considered knowledge acquisition, it could also be argued that knowledge must first be acquired from the expert to be able to know what is "applicable."

With the use of knowledge-based systems becoming more prevalent, the "art" of knowledge acquisition has begun the long journey toward becoming a science. Even so, the literature contains numerous methods for acquiring knowledge. According to Hoffman, all methods for knowledge acquisition fall into one of the five categories shown in Figure 6.

Figure 7 illustrates some of the advantages and disadvantages associated with each of the knowledge acquisition methods. While each method can be useful, only the Method of Familiar Tasks and Structured

METHOD CATEGORY	DESCRIPTION
METHOD OF "FAMILIAR" TASKS	Analysis of the tasks that the expert usually performs.
STRUCTURED AND UNSTRUCTURED INTERVIEWS	The expert is queried with regard to knowledge of facts and procedures.
LIMITED INFORMATION TASKS	A familiar task is performed, but the expert is not given certain information that is typically available.
CONSTRAINED PROCESSING TASKS	A familiar task is performed, but the expert must do so under time or other constraints.
METHOD OF "TOUGH CASES"	Analysis of a familiar task that is conducted for a set of data that presents a "tough case" for the expert.

(Hoffman, 1987:54)

Figure 6

Knowledge Acquisition Methods

and Unstructured Interviews will be discussed here since these methods proved to be pertinent to this research effort.

The method of familiar tasks consists of observing the expert, on a non-interference basis, while he solves a typical problem in the problem domain. If it is not feasible or worthwhile to observe the expert performing tasks (for example, just observing an expert write paragraphs of an acquisition strategy does not provide much insight into the knowledge or reasoning being used), a slight variation can be used. The expert can identify several "typical cases" that have already been solved, and talk through the approach to each case and the reasoning, facts, and

METHOD	ADVANTAGES	DISADVANTAGES
ANALYSIS OF FAMILIAR TASKS	The expert feels comfortable comfortable	Can be fairly time-consuming
INTERVIEWS	For a first- and second-pass at a data base, it can generate much information.	Typically very time-consuming.
LIMITED INFORMATION TASKS	Can be tailored to extract information on selected subdomains of knowledge.	Expert feels uncomfortable and is hesitant to make judgments.
CONSTRAINED PROCESSING TASKS	Can be tailored to extract information on selected subdomains of knowledge, or on the expert's strategies	Expert feels uncomfortable and is hesitant to make judgments.
ANALYSIS OF "TOUGH CASES"	Can yield information about refined reasoning	Occur unpredictably, the knowledge engineer may not be present.

(Hoffman 1987:58)

Figure 7

Advantages and Disadvantages
of Knowledge Acquisition Methods

assumptions used in arriving at a solution. Having the expert perform or talk through typical cases is an excellent way to learn about what types of knowledge and skills are used to solve problems. If this method is coupled with the research of applicable manuals and books, the knowledge gained can help generate a very good "first pass" at a knowledge base. (Harmon and King, 1987:202; Hoffman, 1987:54; Prerau, 1987:46-47; Waterman, 1986:156-159)

The unstructured interview is basically a method in which the interviewer (knowledge engineer) asks spontaneous questions as the expert performs or talks through a task. This method can be very

valuable if it is used as a follow-up to the method of familiar tasks. The assumption here is that the knowledge engineer has used the previous session, including research of written material, to become somewhat "trained" in the problem domain. Thus, the knowledge engineer knows more about what to look for, and what gaps exist in the knowledge from previous sessions, and can get more out of the session by asking questions "spontaneously." The unstructured interview can be valuable for adding information to, and closing gaps in, a first pass knowledge base developed with the familiar tasks method. (Hoffman, 1987:54-56)

The structured interview can be thought of as a follow-on to either or both of the previously discussed methods. In the structured interview, the knowledge engineer asks the expert to review the first pass knowledge base one item at a time to: validate the item; delete the item; restructure ("fine-tune") the item; or add supplemental items. In this manner the entire knowledge base is reviewed and critiqued by the expert. This is analogous to going back to the expert and saying "Here is what I think you said. Is this what you meant?" This feedback to the expert is critical, and can go a long way toward ensuring the "correctness" of a knowledge base before the knowledge base implementation is even begun. The result of the structured interview, if used after one or both of the other methods, is a second pass knowledge base. (Hoffman, 1987:56; LeClair, 1985:184-186; McGraw and Seale, 1987:21; Prerau, 1987:48-50; Waterman 1986:158-161)

The method of familiar tasks, followed by unstructured and structured interviews, was the scenario used to acquire expert knowledge for the prototype knowledge-based system developed in this research. (This is discussed in more detail in Chapter 4.) Methods in the other

three categories proposed by Hoffman were not used and were not appropriate for use in the development of this particular prototype. As implied by Figures 6 & 7, the last three categories of methods are most efficient when the knowledge base has become very large and it is necessary to elicit knowledge for some specific subdomain of knowledge or reasoning. No matter which method or methods for knowledge acquisition are used, there are numerous considerations that must be kept in mind throughout the process. Many of these have to do with the complicated process of human interaction and communication. Although some are quite obvious, the principal considerations followed during this research effort are worth at least a brief mention.

Before even beginning knowledge acquisition, the knowledge engineer must consider where the knowledge comes from. Possible sources of knowledge are books, experts, direct experience, or any combination thereof. The answer to this question will help guide what knowledge acquisition method(s) to use. During the knowledge acquisition process, the knowledge engineer should do his best to put the expert at ease. Many times this will involve a detailed explanation about what the knowledge-based system will be used for, what it can be expected to do, and other details of the overall effort. If the expert knows exactly how he fits in, it will help relieve confusion, apprehension, and perhaps even hostility toward knowledge-based systems. Since the expert is human, another consideration to be followed is to keep each knowledge acquisition session relatively short. While the knowledge engineer (especially a time-constrained graduate student) may be eager to conduct marathon sessions, the expert may get turned-off to the whole project if forced to endure such sessions. In addition, the quality and

completeness of the knowledge and reasoning tends to degrade after more than 2 or 3 hours of continued knowledge acquisition. (Hall and Bandler, 1985:509-510; Tanimoto, 1987:286)

Although the knowledge engineer should guide the sessions, especially the interviews, care must be taken not to "lead the witness" into a particular interpretation of the reasoning or knowledge. Worse yet, the knowledge engineer must never try to force a particular interpretation onto the expert or "aggressively debate" the expert's interpretation. On the other hand, the knowledge engineer should not believe *everything* the expert says just because he says it. The validity of lines of reasoning or use of particular pieces of knowledge should not only be asserted by the expert, but also demonstrated by the expert during problem solving. (Roland, 1986:61-62; Waterman, 1986:154)

It should be obvious by now that successful knowledge acquisition involves many of the same qualities as needed for effective interpersonal communication. Tact, patience, tolerance, intelligence, and a good sense of humor are invaluable. Excellent communications skills and a quick pencil will also help a great deal in developing a "correct" knowledge base.

Knowledge Acquisition With More Than One Expert

Up to this point, discussion regarding knowledge acquisition has assumed that only one expert was available (or at least that only one was used). It is frequently the case however, that the potential exists for more that one expert to contribute to knowledge base development. As knowledge-based systems become larger and even more complex it is likely that the knowledge from several experts will be required (and

desired) for the knowledge base. In fact, relying on a single expert to provide all the knowledge for a knowledge-based system has been proposed as a "pitfall" to avoid (Mittal and Dym, 1985:32). Even so, the problems associated with acquiring and integrating knowledge from more that one major expert per knowledge base have only recently begun being addressed. Consider the following comments referencing knowledge base development utilizing knowledge from more than one expert:

> (under the heading "Summary of the State-of-the-Art) Single expert as a 'knowledge czar.' We are currently limited in our ability to maintain consistency among overlapping items in the knowledge base. Therefore, though it is desirable for several experts to contribute, one expert must maintain control to insure the quality of the data base.
>
> (Gevarter, 1983:203)

> . . . in general, most large-scale systems development efforts have had difficulty incorporating the knowledge of more than one major expert.
>
> (Harmon and King, 1985:199)

The primary problem of using multiple experts when developing a knowledge base is integration of the knowledge. Two problem scenarios exist when more than one expert is involved in contributing to the knowledge base: (1) the combining of expertise from several experts in the same domain, or (2) the combining of expertise from several experts in different domains. Following the convention used by LeClair (LeClair, 1985), the problem of integrating knowledge from more than one expert in the same domain will be called the multiexpert problem; the problem of combining knowledge from more than one expert in different domains will be called the multiple expert problem.

The multiple expert problem has been addressed in some large-scale, knowledge-based systems in use today. For example, HEARSAY and PROSPECTOR handled the multiple expert problem by utilizing several

different knowledge bases, thereby allowing each expert (and his area of expertise) to be concerned with solving only one part of the overall problem (Harmon and King, 1985:200). This is a very effective strategy when the problem to be solved can be broken down into separate sub-problems, with each sub-problem requiring a separate area of expertise (like acquisition strategy paragraphs, with one requiring contracting expertise, one requiring logistic expertise, and so on).

The major problem in the multiexpert situation (several experts contributing knowledge in the same domain) is that there is a potential for the knowledge and lines of reasoning to conflict. Since each expert is an individual, the more experts involved in the knowledge acquisition process (each supplying his own complex solution strategy), the more likely it is that one solution may conflict with another. The current literature revealed two basic approaches for dealing with this problem: (1) reach a consensus among the experts, or (2) allow multiple lines of reasoning to exist in the same knowledge base. Both of these approaches will be briefly discussed here.

In literature that addressed the multiexpert problem, most approaches ended up obtaining a consensus among the experts in some way. McGraw and Seale (Texas Instruments AI Lab) have addressed the question of whether to elicit knowledge from the experts in small groups or individually, and proposed three techniques for group knowledge acquisition (McGraw and Seale, 1987). All three techniques (brainstorming, consensus decision-making, and the nominal group technique) eventually end up by selecting the "best" solution to the problem; that is, a consensus must be reached.

Boose (Boeing AI Center) developed a system called ETS (Expertise

Transfer System) for use in the automated elicitation of expertise and as an aid in reaching consensus among multiple experts in the same domain (Boose, 1985:461-466). Each expert interacts with ETS to produce a "rating grid" that embodies his knowledge regarding the problem of interest. When all experts have created grids, ETS combines the knowledge captured in the grids into rules for a knowledge-based system. The capability also exists for ETS to produce a summary of all the differences between experts, and this summary can then be used to guide a "structured negotiation" session to resolve conflicts between the experts grids. Consensus (to a large degree) must be reached or the system performance will be erratic. Boose has also developed NeoETS to construct multi-dimensional grids (Boose and Bradshaw, 1986:34-45), and since Boose began his work more than 500 knowledge-based systems have been generated by ETS and NeoETS at the Boeing Company with a typical prototype now taking less that two hours to build (Kitto and Boose, 1986:53).

The approach to allow multiple lines of reasoning to exist in the same knowledge base was developed by Steven LeClair, an Air Force Officer, as part of his doctoral dissertation at Arizona State University (LeClair, 1985). LeClair's approach to solving the multi-expert problem is based on the following:

(1) The assumption that the decision-maker (user of the expert system) is probably more of a "master" of the situation-at-hand than any one of the experts. This being the case, he could add knowledge about the current problem context to pick from several solutions if there is conflict among lines of reasoning (and would rather have that choice).

(2) If the system could get feedback from the user about what previous advice was "good" and what was "bad," a system could be created that would "learn" to "forget" bad lines of reasoning and develop new rules of its own about the good lines of reasoning, thus becoming "more expert" than any of the experts.

LeClair considered the multiexpert problem to be simply a problem of unwanted interaction between multiple lines of reasoning resulting from conflicting use of assertions. By developing a method to eliminate this "unwanted interaction," multiple lines of reasoning were permitted in the same knowledge base. He also implemented a conflict resolution mechanism to recognize when conflicting solutions are reached. If conflicting solutions are reached, the system asks the user if he wants the system to recommend a line of reasoning. If the answer is yes, the user is asked to choose which of several presented characteristics is most important to the current problem. The system then "intelligently" selects the solution from the line of reasoning most applicable to the user's current problem. (In addition, LeClair also implemented a mechanism to use a "memory" and fuzzy logic to get feedback from users on previous advice and thereby "learn" which lines of reasoning were "good" or "bad." The system was actually able to take advantage of the conflict in the knowledge base, and "forget" bad lines of reasoning and develop new rules on its own to enable more efficient use of good lines of reasoning.)

Within the acquisition strategy problem domain, the potential exists for both the multiple expert (more than one expert, different domains) and multiexpert (more than one expert, same domain) problems to exist. Since an acquisition strategy addresses many different areas of

expertise (ie., contracting, logistics, testing, and others), several experts are required to participate in the development of the knowledge base with each contributing to a separate "domain" of knowledge. The fact that each document is reviewed at higher levels within the organization before being approved implies that there are "experts" that develop the acquisition strategy and "experts" that review the acquisition strategy. Thus, if the multiexpert problem can be successfully addressed (probably through consensus), the knowledge-based system will provide the added benefit of reducing or eliminating "conflict" during the review process.

As it turned out, the availability of experts during the knowledge acquisition phase of this research (coupled with the scope of the prototype system) dictated that only one expert be used for each "knowledge domain" implemented in the prototype system (multiple expert scenario). The strategy of using several knowledge bases to partition the different areas of expertise worked very well, and provided several benefits as discussed in Chapters IV and V. It is clear, however, that expansion of the prototype into a complete system will have to address the multiexpert problem.

Related Research

The review of current literature did not identify the existence of any knowledge-based (or even "conventional") systems that assist in the generation of acquisition strategies. While several systems do exist to generate various program management and acquisition documents, most of these do not take advantage of the knowledge-based system approach. The Program Manager's Support System (PMSS) itself is an example of this.

The PMSS mainly uses "automated checklists" to help the user generate program management documents. Another example of this type system is the Computer Generated Acquisition Documents System (CGADS). CGADS is a component of the Air Force System Command's Electronic Systems Division Management Information System (AF Systems Command, 1985). CGADS is able to produce textual output through the use of a data base containing text for every possible answer that the user can give to a CGADS question. It also contains "action messages" which tell the user what pieces of information must still be added to the document. While it could be argued that the system contains "knowledge" to a degree, this knowledge is not organized or represented in such a way that the system can "reason" about known facts and deduce appropriate responses or other facts. However, because the system does contain up-to-date references and gives a checklist to guide the user through all tasks that must be completed, it is of significant benefit to the user (as are the PMSS programs).

An example of a knowledge-based system developed to explore the application of artificial intelligence technology to the acquisition environment is the Acquisition Expert System (AES). The AES was developed by The Analytic Science Corporation (TASC) for the Navy Materiel Command (TASC, 1985). The system contains a knowledge base of frames and production rules, and a set of general purpose knowledge base development tools for use in extending the knowledge. The AES was "not intended to be an operational system, but rather a technology demonstration" for evaluating knowledge-based system application to acquisition problems (TASC, 1985:ii). As such, the system is limited to analyzing the Milestone II decision (enter Full Scale Engineering Development

(FSED)) for a tactical missile development. It does not attempt to generate a document, but assists the program manager in addressing concerns in the FSED acquisition phase. The AES was sponsored in part by the Defense System Management College, and is designed to be (conceptually) compatible with the Procurement Strategy Model (PSM), a part of the PMSS.

One other knowledge-based system relevant to this research was identified in the literature; the Knowledge-Based editor in Emacs (KBEmacs). KBEmacs is a system developed to assist programmers in constructing computer programs. While KBEmacs has nothing to do with the acquisition environment it does produce a near-text output (computer code). KBEmacs is also relevant to this research by the fact that the system is designed to be a "knowledgeable assistant" to the user. The goals and ideas behind KBEmacs are very similar to those behind this research: "When it is not possible to construct a fully automatic system for a task, it is, nevertheless, often possible to construct a system that can assist" in the task. Additionally, the "assistant approach can lead to important insights into how to construct a fully automatic system." The primary thrust of KBEmacs was to be a research experiment, and its developers consider it a success as such and its development typical of "other systems that have successfully applied AI techniques." (Waters, 1986:47-56)

Current research in the area of hypertext could prove to be highly useful in generating textual documents. Briefly, hypertext can be described as a "computer-based medium for thinking and communication" (Conklin, 1987:32). The concept underlying hypertext consists of associating objects in a database with windows on the screen. The objects

can be "text nodes," or in a more general sense, ideas. The windows on the screen and the database objects are connected via "links." Hypertext links two points either by reference (from a source to a destination) or by organization (a parent object with its children, much like a semantic net). The links can connect a comment to the text it references (for example, the rationale for a particular section in an acquisition strategy) or can connect two pieces of text, or a piece of text and all the text before it, in a hierarchical fashion (for example, a particular paragraph in an acquisition strategy with the paragraphs that proceed it. (Conklin, 1987:17-40).

NoteCards, developed at the Xerox Palo Alto Research Center as a research project to study hypertext, is probably the best known hypertext implementation. Using NoteCards to develop a document is much like using regular 3x5 notecards with bits of information on them except that the "notecards" are in "files," and they can be "linked" together to form all or part of the document. The biggest benefit is that the manipulation of the notecards and the notecard files is computerized. NoteCards runs on Xerox D-series Lisp machines which use high resolution screens, thus enabling display of the windows and graphical links in very high resolution. (Conklin, 1987:27-28)

Hypertext is not without some disadvantages. Current implementations have several deficiencies, including slowness. The complex nature of the organization of information can also cause the user to become "lost" and become tired of the "mental overhead." The potential operational advantages, however, seem to be far greater. These advantages include (but are not limited to): (1) the ability to link segments of text together in many ways, allowing one document to serve many

purposes; (2) the ability to avoid duplication of ideas by allowing segments of text to be referenced from many places; and (3) the ability to keep information consistent since the text carries the (embedded) references with it when it is moved. (Conklin, 1987:38-40)

The ability to treat ideas as objects suggests a tremendous potential for providing assistance in the creative task of developing a document. The implications of combining hypertext with knowledge-based systems is discussed in Chapter VI.

Summary

In this chapter, information from current literature applicable to setting the stage for this research effort was reviewed. First, the definition of knowledge-based systems was discussed; then knowledge-based systems and conventional programs were compared; guidelines for when to use knowledge-based systems was presented; knowledge acquisition was discussed; and finally, research related to this effort was identified.

It should be apparent from the discussion that knowledge-based systems can be extremely beneficial in assisting a user with tasks that require significant amounts of expertise. While knowledge-based system techniques can help solve problems not amenable to traditional programming solutions, they also provide the benefits of a friendly user interface and rapid prototyping. Knowledge-based systems can be helpful in situations where expertise is constantly being lost due to personnel changes, where experts are scarce, or when experts are needed in many locations.

All of the above implies that the knowledge-based system approach

is appropriate for application to the problem of generating acquisition strategies. Indeed, it was precisely this conclusion that spawned this research effort. In the next chapter, the detailed problem assessment is presented that was used to verify the initial premise that the knowledge-based approach was appropriate, and that was used to formulate a conceptual solution.

III. PROBLEM ASSESSMENT

In this chapter, the information gained during the problem assessment phase of the research is presented. The following methodology was used to collect the information:

- A Problem Assessment Checklist was developed (Appendix A) and was furnished to BELVOIR points of contact. (The format and content of the checklist was patterned very closely after a checklist developed by Teknowledge, Inc.)
- Approximately two weeks after mailing the checklist, a visit was made to Fort Belvoir to go through the questions with BELVOIR personnel.
- In addition to addressing the questions in the checklist, a "typical" acquisition strategy (Appendix B) was examined and the methodology for development of the document was discussed.

The BELVOIR representatives that provided the information during this phase were Mr. Dan Causey, Chief, Advanced Systems Logistics Division, Advanced System Concepts Directorate, and Ms. Tammie Vogler, Project Engineer, Advanced Systems Logistics Division. Both of these individuals have had several years experience being a project engineer, writing acquisition strategies, and reviewing acquisition strategies written by others. In addition, both are pursuing advanced degrees in computer science and have a good grasp of knowledge-based system techniques and capabilities. The interview to address items in the checklist and to discuss a representative acquisition strategy lasted approximately three hours.

The material presented in this chapter is organized in the

following order: first, the problem definition is presented, followed by sections addressing the current practice for developing an acquisition strategy, the characterization of the problem that the expert is solving, the characterization of a successful knowledge system, a definition of a prototype system, possible evolution of the prototype system, and resources required to develop the prototype system. Finally, the methodology that a typical project engineer would have used to develop the acquisition strategy for the LAMP-H program is presented.

Problem Definition

The project engineer assigned to a program is responsible for developing the acquisition strategy for that program. As previously stated, the acquisition strategy is one of the most important program documents and is required for all materiel acquisition programs regardless of dollar amount. The primary purpose of the acquisition strategy is to document the general concepts that serve as direction for the program throughout the entire acquisition cycle. The problem that a project engineer faces is that information required for an acquisition strategy is too diverse for one person to formulate. While the project engineer has detailed knowledge about the program itself, he usually does not know enough about the variety of topics that must be addressed in an acquisition strategy to write it alone. For example, the detailed information for paragraph 2, Contracting Strategy, can best be provided by contracting personnel. Input for paragraph 4, Supportability, is generally required from personnel knowledgeable in the Integrated Logistics Support area. Similarly, personnel from the BELVOIR drawing vault, Test and Evaluation division, Cost Estimation division, Safety

division, Human Engineering Lab (HEL), Standardization division, and Survivability division would be involved in paragraphs 5, 6, 7, 9, 10, 11, and 12, respectively. The ideal situation would be for the project engineer to meet with representatives from all the required matrix support areas as the appropriate portion of the document is written, but time requirements and coordination problems make this solution impractical.

Confronted with having to develop an acquisition strategy, the typical project engineer will usually try to write the entire document himself and then staff it through the various matrix support areas for comment. The biggest impact to the organization caused by the "diversity" problem is the time wasted in the "write-review-rewrite" cycle. Ideally, the cycle would only be completed once. For several reasons, however, that is usually not the case. For example, the matrix support personnel may have incomplete knowledge about some aspect of the program, and thus (inadvertently) provide only part of the information and corrections necessary. During the second review, when more information has been added to the document, previous input may be changed or additional input added, or both. In addition, changes made to areas of the document based on the previous review(s) may now cause changes in other areas to be appropriate. Also, on subsequent reviews a person may read the document and have a slightly different interpretation of the situation than he had before, or notice something that was not caught earlier, and initiate another change (possibly causing the ripple effect to start all over again). Since so many areas and topics must be addressed in an acquisition strategy, it is also possible for the document to get through one or more reviews before someone "discovers" that

a critical or required aspect of the program has not been addressed. Multiple iterations of the "write-review-rewrite" cycle may occur within BELVOIR before the document is considered "correct" enough to send to the next higher organization for review, which could then start the process again. In addition to wasting time, emotions from both the project engineer and the reviewers may enter into the situation after several rewrites, causing conflict within the organization.

It is also typically the case that the project engineer has seldom had prior experience in preparing acquisition strategies. Thus, time must be spent searching for the most recent regulations, handbooks, and other documentation so that he is aware of all requirements and guidance. This can add a significant amount of time to the initial development of the document as well as to the review process.

In addition to the problem of the diverse areas that must be addressed in an acquisition strategy, a second problem for BELVOIR is the large number of programs for which they are responsible. At any one time, up to 200 programs may be in various stages of the procurement life cycle. Since each program (usually) has a different project engineer, the acquisition strategy for each program is developed by a different person. One impact of the large number of programs for which BELVOIR is responsible is that there are many different people competing for the expertise of the limited matrix support offices. A program just getting started (developing an acquisition strategy) must utilize the knowledge of the same people involved in supporting all the programs already underway. An additional impact is the inconsistency in the acquisition strategies produced by all the different project engineers. Another critical impact is that, since many people are involved in

developing and reviewing acquisition strategies, it is difficult to ensure that everyone has the latest requirements and guidance. This is complicated by the fact that acquisition regulations change frequently.

The time delays caused by the review cycle, the search for applicable guidance and regulations, competition for resources, and any rewrites caused by inconsistency or use of obsolete data ultimately affects the progress of the program. The effects can range from a delay in type classification of the item, to adverse impacts on program funding, to delays in equipment delivery. Any proposed solution to the above problems should:

- provide the project engineer with the ability to call upon knowledge from personnel in each of the support areas during the development of the acquisition strategy;
- provide the project engineer with the ability to take advantage of experiences of other project engineers that have already gone through the process of developing an acquisition strategy;
- incorporate knowledge from the reviewers;
- assist in providing consistency among acquisition strategies;
- facilitate the dissemination of changes in acquisition regulations and guidance;
- capture knowledge of the approvers; and
- provide a consistent framework within the dynamic decision environment.

Current Practice

As mentioned in the previous section, the current practice in developing an acquisition strategy is for the project engineer to write

the best document he possibly can (in at least semi-isolation) and then staff it through the support areas for comment. Many times the project engineer will use one or more successful acquisition strategies for similar programs as an example. However, since no history is kept of why certain aspects of a program are addressed in the manner they are within an acquisition strategy, this approach may provide little help. The project engineer essentially tries to figure out what must be said to get an acquisition strategy approved for his program. This would probably be an acceptable strategy if only common sense was needed to meet the requirements. The old saying "there is a right way, a wrong way, and the way we do it here" may apply to this situation, however. There are certain required elements that must be addressed by every program, that may or may not be able to be deduced by a person using common sense about what is "reasonable" to discuss. Within the required items (the paragraphs), information presented is largely based on the procurement type. This information is, to large degree, contained in various regulations, handbooks, and local documents. But based on other factors, such as the personality of the current approval authority, past experiences with a particular agency, or the political climate, the "experts" in each of the support areas may have unwritten "rules-of-tnumb" that they use to develop and critique the portions of the acquisition strategy that fall within their area of expertise. So, the overall ability to develop an acquisition strategy has little to do with "common sense", and the specialized knowledge required is split between manuals and personal heuristics.

It is estimated (by BELVOIR representatives) that a project engineer experienced in the development of acquisition strategies could

probably produce a draft document in a week or less. In contrast, an inexperienced project engineer would probably take several weeks or more to produce the same document. This illustrates a significant time disparity, and there are currently no active training programs within BELVOIR aimed at closing this gap.

The criteria for determining that an acquisition strategy has "failed" is when a major rewrite has to be done. This can be caused by a rejection from the program office, the matrix support offices, or the Material Acquisition Review Council (or Board, whichever is appropriate for the particular program). A "failure" can also occur when the acquisition strategy is reviewed at each milestone within the program lifecycle. Typically, a "failure" occurs more than once for every acquisition strategy. It is recognized that this criteria for failure is severe, and any approach, knowledge-based or otherwise, could probably not eliminate all failures. However, the time to "correct" a failure could be substantially reduced. For example, it usually takes up to month to get the in-house review cycle completed (just involving the project engineer, program office, and matrix support offices).

A knowledge-based system would probably need to follow the same general approach for developing an acquisition strategy that is used now. That is, produce an initial draft and staff it for review. However, the key difference is that a knowledge-based system should take advantage of the expertise of the personnel in the support areas and approval areas when assisting the project engineer to develop the initial acquisition strategy.

Characterization of the Problem that the Expert is Solving

In producing an acquisition strategy, the "solution" is definitely one of "satisfice" versus one of "optimize." That is, there is a set of constraints (required items) that must be satisfied (addressed sufficiently). Any acquisition strategy that meets the goal of getting approved is a solution. There may be a large number of possible "solutions" for any one program, but no requirement exists to prove that a solution is "optimal" (in fact it is most likely impossible to do so). The set of solutions is largely pre-defined by the nature of the program but still quite large. When developing the acquisition strategy, the potential solutions can be efficiently "pruned" by examining particular characteristics of the program, such as the current phase of the life cycle the program is in or the contract type.

The input used to develop an acquisition strategy comes mainly from documents reflecting the equipment requirements, capabilities, and other basic characteristics of the item to be procured. All this information is largely correct at any particular point in the program life, but is highly dynamic. For example, funding levels and cost data for the equipment typically change throughout the life of the program. It is for this reason that an acquisition strategy is a "living document." The dynamic nature of the data, therefore, must be ignored to produce the document at all, and it is just a fact of life that the acquisition strategy must be continually updated.

The reasoning process followed by an "expert" developing an acquisition strategy is basically the same for any program. First, initial data must be gathered (if not already known) to establish the characteristics of the program. Then, given that the dynamic data is

correct at this particular point in time, the various required items of the acquisition strategy are addressed using regulatory and local guidance, and any "rules-of-thumb" based on previous experience in developing an acquisition strategy and getting it approved.

Characterization of a Successful Knowledge System

The main characteristic that a successful complete knowledge-based system should have is the capability to propose an initial draft acquisition strategy addressing all 13 required paragraphs. The system would have to be able to propose acquisition strategies for Non-Developmental Item (NDI) procurements as well as Research and Development (R&D) procurements. The system would need to be able to ask a user (the project engineer) questions about the program and generate text to an output file, with the resulting document considered an "acceptable" first draft. Zero "incorrect" solutions is recognized as an impossibility. The capability to edit the document should also be provided. Response time and any particular graphics capability are not concerns.

The specific environment within BELVOIR also dictates several characteristics. To allow for easy integration into the organization, the system would have to be microcomputer-based. Specifically, it should be able to be used on the Zenith 248 microcomputers currently available within BELVOIR. Since all system maintenance would be performed by personnel within Mr. Causey's office, ease of maintenance should be "designed in." Maintenance includes not only "code" maintenance, but also the functions of updating and changing the knowledge base, as well as allowing for changes in the form of the document and the items addressed (based on changes in acquisition regulations and

guidance). The system should also have the capability to integrate into existing data bases (and some now under development) at BELVOIR. Currently, however, all the data bases are not integrated themselves, although a project is underway to solve this problem.

The benefits that BELVOIR expects from a successful system are:

- Ultimately, reduce the review process among BELVOIR offices to one cycle;
- Take advantage of previous experience in getting acquisition strategies approved;
- Reduce the competition for the resources in the support offices;
- Preserve knowledge of personnel that leave/retire;
- Improve consistency among documents.

BELVOIR recognizes that coordinating the availability of all required "experts" for the development of a knowledge-based system would be difficult. It is felt, however, that the benefits would outweigh the inconvenience. Similar thinking applies to the process of integrating the system into the organization; it would not be a simple task, but the benefits would go a long way toward eliminating any reluctance.

Defining a Prototype

A successful prototype system would not have to have all the capabilities of the complete system. To demonstrate the effectiveness of applying knowledge-based system techniques to the problem of developing an acquisition strategy, the prototype would only need to produce a document addressing one or two of the 13 paragraphs. Mr. Causey originally proposed that the prototype system could either address one or two paragraphs in great detail or generate a simplified, general

version of the entire document. Upon reflection, he determined that addressing one or two paragraphs in detail would be of more immediate benefit to the organization. By providing the capability to develop one or two paragraphs fully, the impact and advantages of a knowledge-based system could be more dramatically demonstrated to organization personnel, and the system could begin to be used to develop these paragraphs. Also, not as many experts would have to be scheduled for availability during the knowledge acquisition phase.

Additionally, the prototype would only need to address NDI procurements to accomplish the goal of demonstrating the effectiveness of knowledge based systems for developing an acquisition strategy. This decision was made by Mr. Causey in light of the time allowed for this research and the anticipated availability of BELVOIR experts. The fact that all BELVOIR procurements are classified as non-major weapon systems and that the NDI approach is applicable to many of them was also considered.

The prototype would need to have most of the other characteristics of the complete system. It should be usable on the Zenith 248 microcomputer and be designed for easy maintenance. It would need to interface with the user to get information about the item to be procured, and it should provide a capability to edit the proposed initial draft. The prototype would not need to interface with existing BELVOIR data bases, however, especially in light of the fact that the project to integrate the data bases among themselves is still ongoing.

An additional characteristic that the prototype would need is the ability for expansion into a complete system. That is, it should be designed so that the capability to address the remaining paragraphs, as

well as R&D procurements, could be added. Table I summarizes the requirements for a successful prototype knowledge-based system as compared to those for a successful complete system.

TABLE I

System Requirement Matrix

REQUIREMENT	PROTOTYPE	COMPLETE SYSTEM
Propose Initial Draft	Yes	Yes
Number of Paragraphs	2 of 13	All
Procurement Type	NDI	NDI and R&D
Z-248 Based	Yes	Yes
Interface with User	Yes	Yes
Allow Document Editing	Yes	Yes
Easily Maintainable	Yes	Yes
Integrate with BELVOIR Databases	No	Yes
Easily Expandable to Complete System	Yes	---

Evolution of the Prototype

As already indicated, the prototype would have the potential for expansion into a complete system. Although it would address only one or

two of the paragraphs of the acquisition strategy, it could be expanded by following the same implementation strategy for the additional required paragraphs. Interfaces into the existing data bases could be integrated into the prototype as appropriate. The environment for the prototype and the complete system would be the same. In addition, the hardware requirements for both systems would basically be the same, except for any disk space required for the extra knowledge added to expand the system. It is also possible that some additional hardware would be needed for the basic system to enable an interface with any data bases.

Resources Required

The resources required for development of the prototype are minimal. While many typical cases would be needed to provide at least one case for each of the types of programs found within BELVOIR, the prototype can be developed using only a few to demonstrate the concept. The target hardware for the prototype, as well as a complete system, is the Zenith 248 microcomputer. Thus, any knowledge-based system developed has to be capable of implementation on the Zenith 248. The only software necessary for system development is an appropriate knowledge-based system building tool (to be determined after all aspects of the problem and the solution requirements are known).

The most important resource required for the prototype, and perhaps the most important and hardest to obtain for any knowledge-based system, are the experts themselves. To obtain the requisite knowledge for all required items in an acquisition strategy, at least one project engineer that is "expert" in the development of acquisition strategies would be

needed, along with at least one expert in each area to be addressed by the system (ie., contracting, logistics, cost estimation, etc.). The difficulty anticipated (by Mr. Causey) with scheduling the required experts serves as a case in point of how a knowledge-based system could be of benefit. Once the expertise from these experts is captured in a system it would not only be readily available for use by all project engineers, but would also reduce some of the workload for the experts thereby allowing them to concentrate on more creative tasks.

LAMP-H Acquisition Strategy

To provide an initial understanding for what an acquisition strategy is and what knowledge is required to develop one, a typical acquisition strategy was discussed during the Problem Assessment phase. The LAMP-H Program acquisition strategy (Appendix B) was chosen by Mr. Causey. It should be noted that the discussion with Mr. Causey was, at this point, aimed only at developing a general understanding, and was not intended to be as detailed as an examination of a typical case would be for knowledge acquisition. Mr. Causey and Ms. Vogler have had considerable experience preparing and reviewing acquisition strategies and provided the paragraph-by-paragraph discussion to be presented here.

Paragraph 1: Program Structure. The first paragraph is usually written entirely from the project engineer's knowledge of the program. Typically, he will get an old acquisition strategy and begin to write. The goal in the first paragraph is to present the program structure while going into as little detail as possible. The first step is to decide what kind of program it is; for example, is it an NDI acquisition or an R&D acquisition? The type of program determines the phases that

the program will go through and what In-Process Reviews (IPRs) will be conducted. The program documents listed in the LAMP-H acquisition strategy are generally the same for most programs. Most NDI programs will include a Pre-Planned Product Improvement (P3I) Program to help shorten the acquisition time. During the development of paragraph 1, the project engineer may consult with the program office in his directorate, or other acquaintances and project engineers that have had experience in preparing an acquisition strategy.

Paragraph 2: Contracting Strategy. The information in this paragraph typically comes entirely from contracting personnel. The contracting strategy is highly dependent upon the program type (NDI or R&D) and which phase of the life cycle the program is in. Technical risk, program objectives, and the quantities of the item that will be acquired also influence the contracting strategy. The contract type selected can also be based on political factors, such as preferences of the current approving authority or pressures from Congress to use certain types of contracts.

Paragraph 3: Tailoring the Acquisition Process. This paragraph is usually written by the project engineer, and may be the toughest part of the entire document. In fact, the project engineer may develop this paragraph after the rest of the acquisition strategy is finished. Input for paragraph 3 comes from regulations, the Engineer's Handbook (a BELVOIR document), and previous examples. The completion times for various program events can be based on several factors, including the program type and how much basic research will be involved in the program.

Paragraph 4: Supportability. Input for this paragraph usually

comes from the Integrated Logistics Support (ILS) personnel. It can be based on regulations, past experience, or both. The thrust of this paragraph is to say what is going to be done, but not the details of how it will actually be accomplished.

Paragraph 5: Manufacturing and Production. The major portion of the information contained in this paragraph comes from BELVOIR drawing vault personnel. For a government design, a Technical Data Package (TDP) is usually required. For the LAMP-H program, this was modified somewhat to include specifications also. Design changes are always controlled by Engineering Change Proposals (ECPs). Value Engineering (VE) is always addressed, either in the form of a VE Plan or VE Procedures. Also in this paragraph the decision must be made to either address the Unit Production Cost and Baseline Cost Estimate or try to get by without them.

Paragraph 6: Test and Evaluation. The main source of information for this paragraph is the Test and Evaluation division. As with other paragraphs, the idea is to give general statements as to what will occur without going into detail. Typically, this paragraph acknowledges that testing will occur in accordance with AR 70-10, that critical test issues are contained in the Independent Evaluation Plan (IEP) and Test and Evaluation Master Plan (TEMP), and that the principles of Continuous Evaluation (CE) will be followed.

Paragraph 7: Cost Growth and Driver. The information in this paragraph comes mainly from the Cost Estimation Division. In the LAMP-H acquisition strategy this paragraph is larger than normal due to the high overall cost and some other unique features of the program.

Paragraph 8: Technical Risks. This paragraph is usually developed by the project engineer. The thrust of this paragraph is to present an argument for why there is a chance for the program to succeed. The information addresses strictly technical risk, and states that it is high, low, or medium and why.

Paragraph 9: Safety and Health; paragraph 10: Soldier-Machine Interface; paragraph 11: Rationalization, Standardization, and Interoperability; paragraph 12: Survivability and Endurance; paragraph 13: Short Term Issues. Paragraphs 9 through 13 are typically standard for every program. If changes are necessary, the major source of input for the change is as follows:

- Paragraph 9: Safety Division
- Paragraph 10: Human Engineering Lab
- Paragraph 11: Standardization Division
- Paragraph 12: Survivability Division
- Paragraph 13: Anyone

To assist in deciding which paragraphs had the most potential payoff and which paragraph(s) would have to be implemented to demonstrate the feasibility of the knowledge-based system approach, Table II was developed. Mr. Causey was asked to rate each paragraph based on difficulty to develop (1 to 5, 5 being the hardest); whether only experts could develop the paragraph well, or no one could develop it well, or anyone could develop it well (Category A, B, or C, respectively); and the estimated time (in hours) it would take the expert to develop the paragraph versus the novice. Based on the ratings, paragraphs 3, 2, 7, and 1 (in that order) are shown as the most difficult to develop the first time. Mr. Causey indicated that implementation of

TABLE II

Acquisition Strategy Paragraph Evaluation

PARAGRAPH	DIFFICULTY	CATEGORY	TIME
1. PROGRAM STRUCTURE	4	A	4/16
2. CONTRACTING STRATEGY	5	A	4/40
3. TAILORING THE ACQUISITION PROCESS	5	A	16/80
4. SUPPORTABILITY	3	A	2/16
5. MANUFACTURING & PRODUCTION	3	A	2/16
6. TEST & EVALUATION	2	A	1/4
7. COST GROWTH & DRIVER	5	A	8/30
8. TECHNICAL RISK	3	A	4/16
9. SAFETY & HEALTH	1	C	1/1
10. SOLDIER-MACHINE INTERFACE	1	C	1/1
11. RELIABILITY, SUPPORTABILITY INTEROPERABILITY	2	C	1/1
12. SURVIVAL & ENDURANCE	2	C	1/2
13. SHORT TERM ISSUES	1	C	1/1

LEGEND: DIFFICULTY -- RATED FROM 1 (easy) TO 5 (hard)

CATEGORY -- A: EXPERTS DO WELL
B: NO ONE DOES WELL
C: ALL DO WELL

TIME -- TIME FOR EXPERT TO DO/TIME FOR NOVICE TO DO (in hours)

RESULTS: Paragraphs that are the most trouble the first time (thus cause most rewrites) are paragraphs 3,2,7,1 (in that order). According to the expert interviewed, the biggest payoff would be with paragraph 3, but the implementation of 1,2,3 or 7 would suffice to sufficiently demonstrate the concept of using a knowledge-based system to solve the problem.

paragraph 3 would probably provide the biggest benefit to the organization since it by far seemed to take the novice longer to develop, but he also indicated that successful implementation of any of the four would serve to prove the applicability of the knowledge-based approach. (While an attempt was made to provide experts from all four areas, subsequent expert availability and qualifications focused efforts on paragraphs 1 and 3.)

Summary

In this chapter, the problems associated with developing an acquisition strategy were defined, the current practice used to develop the document was reviewed, and characteristics of the problem and characteristics of a successful knowledge-based system were presented. In addition, a prototype system was defined, along with considerations for system expansion and the resources required to develop the prototype system, and a "typical" acquisition strategy was examined. Finally, the evaluation of acquisition strategy paragraphs that focused implementation efforts was presented.

From the problem assessment, it is clear that the knowledge-based approach is a reasonable method to consider for application to the problem of developing acquisition strategies. The requirements for knowledge-based system development, illustrated by Figure 3 in Chapter 2, are all sufficiently met. (It is assumed that the "experts" will be as articulate as Mr. Causey and Ms. Vogler, thus meeting the third requirement.) In addition, the competition for the personnel resources in the matrix support offices meets the "human expertise scarce" criterion in Figure 4, thus making knowledge-based system development justified.

Justification is also provided by the fact that a potential for high payoff exists and that human expertise is frequently lost due to personnel retiring or being reassigned. The criteria shown in Figure 5 are also met, thereby making knowledge-based system development appropriate.

Most of the other considerations presented in Chapter II for when to use the knowledge-based system approach are also supported. Information gained in the problem assessment phase indicates that BELVOIR management is willing to support the development of the system and does not have unrealistic expectations about system performance. The exact degree of user acceptance or reluctance is not known, but information suggests that the users (project engineers) would welcome assistance in the task of developing acquisition strategies. The problem assessment phase has shown that test cases are available. Conventional solutions to the problem have been considered, but due to the use of "rules-of-thumb" in the generation of acquisition strategies and the requirement to frequently change "knowledge" in the system as the regulations and guidance change, the conventional (algorithmic approach was discarded.

IV. DESIGN AND IMPLEMENTATION

Introduction

Before discussing aspects of the prototype system design and implementation, it is appropriate to briefly discuss phases in the development of a knowledge-based system and outline the approach used in this research effort. While various steps are proposed throughout the literature for knowledge-based system development, most fit the following general form: (1) Identify the Problem, (2) Develop a Prototype, (3) Expand the Prototype, (4) Evaluate the System, (5) Deliver the System, and (6) Maintain the System (Harmon and King, 1985:197-207; Citrenbaum and others, 1987:32-33). These steps are considered "typical" for a large knowledge-based system that will ultimately be delivered to a user community for long-term usage. Since the system developed in this research is a prototype, the phases of large scale knowledge-based system development covered by this effort are (1) and (2). However, since the prototype was actually the end product of this effort, it was considered t - "system to be delivered" and the prototype development was broken int phases co·responding roughly to those of large scale knowledge-based system development. Thus, the approach used in this effort is mapped to the general form for knowledge-based system development as follows:

(1) Identify the Problem. This corresponds to the problem assessment activities covered in Chapter III.

(2) Develop a (Pre-) Prototype. For this step, an extremely small, simple system was constructed using a conceptual design based on information gained in the problem assessment phase; a pre-prototype.

(3) Expand the (Pre-) Prototype. Activities contained in this step include the knowledge acquisition phase, the detailed design of the

prototype, the tool selection process, and the implementation of the prototype with the selected tool.

(4) Evaluate the System. This step directly corresponds to the development sequence used for the prototype and, thus, covers the prototype evaluation.

(5) Deliver the System. Actions in this step include optimizing the speed of the system, customizing the user interface characteristics based on the evaluation, and "fine-tuning" the knowledge-bases based on the evaluation.

(6) Maintain the System. This step was not included within the scope of this research (but its implications were considered in the prototype design).

Information presented in this chapter is organized as follows: First, the development of the pre-prototype is discussed. Then sections addressing the knowledge acquisition process, the detailed design, and the tool selection process are presented. Finally, the prototype knowledge-based system implementation is examined.

Pre-Prototype Development

From information gained during the problem assessment phase, it was possible to develop a conceptual model for the prototype system. The problem assessment indicated that the major problem encountered during the development of an acquisition strategy (especially by a novice) was the diverse nature of the knowledge needed by the project engineer to write the document. This knowledge is continued in manuals, written guidance, and in the heads of experts in the various matrix support areas (logistics, contracts, etc.) as well as in the heads of experienced project engineers and the acquisition strategy reviewers and approvers. The challenge was to make all this knowledge available to the project engineer in a manner that would provide him assistance during the development of the document. This analysis suggested that an

"intelligent assistant" (in the form of a knowledge-based system) with access to all the required expertise could provide the desired assistance to the project engineer. Figure 8 graphically represents this concept.

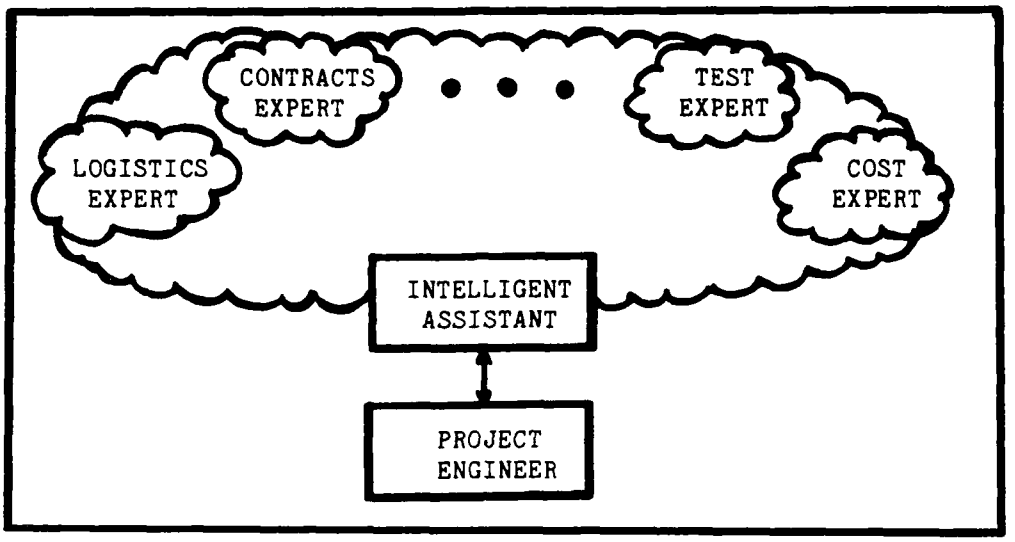


Figure 8

Conceptual Design

Based on the concept described in figure 8, an extremely limited, rudimentary system was developed. The purpose of this "pre-prototype" was not only to test the concept, but also to facilitate the knowledge acquisition process by enabling a small demonstration for the experts. The thought was that many of the experts might not be familiar with how knowledge-based systems work or what they are capable of, and a demonstration would provide a baseline of understanding for the experts, help bound their expectations of the system, and stimulate discussion about user interface considerations.

The pre-prototype was comprised of one rule base with 10 simple rules, embedded in "conventional" procedural code that controlled user interface mechanisms and invoked the "consultation" with the rule base. The rules dealt with a grossly simplified selection of the proper contract type for an acquisition program. Rule content was based on information obtained from regulations, and rule "correctness" was not of great concern.

Based on information known at this point about the problem domain, the knowledge-based system development tool Guru was selected to use for the pre-prototype implementation. Guru is an easy tool to learn to how use, and the problem characteristics seemed to suggest that Guru might be appropriate for the prototype development to be done later. (The tool selection process used to pick the tool for the prototype implementation is presented later in this chapter.)

The construction of the pre-prototype accomplished several things. First, it stimulated the author's thinking in the area of rule interaction and design. It also reinforced ideas about how detailed the knowledge has to be to permit representation in a knowledge base. It proved to be difficult to represent the "knowledge" taken from the regulations without the benefit of knowing how an expert reasons and uses particular facts. The experience of trying to develop a knowledge base without complete information about how the knowledge is used proved to be invaluable by helping prevent this type of mistake during the knowledge acquisition process.

The development of the pre-prototype also helped increase confidence in the conceptual model. Having an "assistant" that could consult appropriate experts as needed (the rule base with knowledge

about contract types) worked very well in the pre-prototype. The reasoning process followed by the system (gathering facts and then asking the "expert" to make a decision based on the facts) seemed to be logical and seemed to follow what the project engineers would like to be able to do when developing an acquisition strategy.

Although the knowledge acquisition phase is examined next, it seems appropriate to discuss the contributions of the pre-prototype during that phase here. Having the capability to demonstrate even a small knowledge-based system (as the pre-prototype was) proved to be an enormous help. The use of a "real" demonstration generated much more enthusiasm from the experts than simply discussing knowledge-based systems in unfamiliar, abstract terms. As each expert saw the demonstration, they immediately began to think of "what-ifs" for user interface mechanisms and for application to their particular area of expertise. By seeing the system ask questions to get facts, and then reason about these facts to produce other facts and recommendations, the experts were able to map this methodology to their own reasoning process. Many times an expert would talk through how he made a particular decision, and would then say "The computer might be able to do this by . . . ." The pre-prototype demonstration seemed to give the expert something to relate to, and helped the expert think more in terms of what facts were needed, and what reasoning was done about the facts, for a particular decision. The development of the pre-prototype was unequivocably worth the time and effort.

Knowledge Acquisition Phase

The next step in the design and implementation of the prototype

knowledge-based system was knowledge acquisition. This phase consisted of identifying and interviewing experts within BELVOIR during a two week timeframe. Based on the acquisition strategy paragraph evaluation (presented in Chapter III), initial meetings with experts in the areas of tailoring the acquisition process (paragraph 3), contracts (paragraph 2), cost estimation (paragraph 7), and in the development of the acquisition program structure (paragraph 1) were solicited. Experts within BELVOIR were identified and available for all areas except for cost estimation, and an expert in the area of integrated logistic support (paragraph 4) was substituted.

Initial meetings were conducted with all four experts on an individual basis (as were all meetings). The initial meetings were used for introductions and to explain the research, the purpose of the research, and get a feel for the experts knowledge and perceptions of knowledge-based systems (if any). The initial meeting was also used to schedule the next meeting with the experts.

Follow-on meetings were conducted with all four experts. During this second meeting, each expert was asked to "talk through" the development of his particular paragraph for a "typical" acquisition strategy that he had previously helped write. In keeping with the method of familiar tasks described in Chapter III, the experts were not interrupted as they "worked." When the expert finished talking through the performance of the task, questions were asked to help clear up any misunderstandings and fill in gaps in the information provided by the expert. A third meeting was scheduled, and the expert was asked to have at least one more "typical case" available for that meeting. In addition, each expert recommended manuals, regulations, and written policy

that was applicable to his particular area for additional research by the author.

It should be noted here that the concerns regarding availability of experts voiced by Mr. Causey during the problem assessment phase were not unfounded. It was a real challenge for the experts to juggle their schedules to make time for continued participation. In fact, sufficient time could not be provided by the logistic and contracts experts, and further meetings with these individuals were unable to be scheduled.

During the third meetings (with the paragraph 3 and paragraph 1 experts), the pre-prototype system was demonstrated with favorable results as previously discussed. Before the expert began talking through another typical case, questions were asked to clear up any misunderstandings and gaps in the data that still existed. The documents recommended by the experts as applicable to their area had been examined since the last meeting, and questions were asked to help tie together the information in those documents with data provided by the experts at the last meeting. Then, the expert was asked to talk through another paragraph development process for a "typical case" acquisition strategy. This process was conducted as an unstructured interview; that is "performance" of the task by the expert was interrupted by questions as necessary. At the completion of the meetings, a fourth (and, due to time constraints, final) meeting was scheduled with each expert.

The fourth meeting was conducted as a structured interview. Between the third and fourth meetings, the knowledge and reasoning used by each expert was put into the form of English-like if-then rules. The interpretation of what facts the expert used to make decisions and what decisions were made based on certain facts that was embodied in the

rules was presented to the expert for comment. It was basically a feedback process using a "this is what was heard; is that what was said" methodology. The result was that some "rules" were changed and some new information was added to provide a more clear interpretation. For the most part, however, the facts and reasoning processes were correct as interpreted. At the end of the meeting, the experts were advised that after the prototype system was designed and implemented they would again be needed for the evaluation phase. This concluded the knowledge acquisition phase.

During the entire knowledge acquisition process, the experts were extremely cooperative and tried their best to provide complete and accurate information. This was true even when their participation caused them to miss lunch breaks or stay at work after the normal duty day (more than once to 7 pm!). Mr. Causey arranged for the author to be provided an account on the INFOMAIL system used within BELVOIR and provided a toll free number so that continued contact could be maintained with the experts.

Although not originally planned as part of the knowledge acquisition process, the opportunity to attend a BELVOIR Materiel Acquisition Review Board (MARB) was very helpful. The MARB is the body that reviews every program within BELVOIR, and either recommends approval or disapproval of the acquisition strategy (approval must occur before the program can proceed). As luck would have it, the acquisition strategy for the program reviewed was questioned in a couple of key areas, and approval was granted contingent upon the concurrence of the experts in those areas, or the document had to be rewritten so that they would concur. (One of the required concurrences was needed from the

expert that had been interviewed from the contracts area; the project engineer had not sought his input because he knew how busy the contracts office was!)

The information gathered during the knowledge acquisition phase was considered complete for the implementation of paragraphs 1 and 3 for Non-Developmental Item (NDI) procurements within BELVOIR. This was appropriate for the scope of this effort, and sufficient for demonstrating that the knowledge-based system approach was applicable to the problem of developing acquisition strategies.

Other details regarding the preparation of acquisition strategy paragraphs was also gained. The reasoning process demonstrated by the experts followed an if-then methodology. The experts first gathered facts about the program, and the reasoned that if certain facts were true, then other facts were true. When enough facts were determined to be true (or false), then a conclusion could be reached. For example, if the item to be procured is commercially available, and if the item is a replacement for another item currently in the inventory, and if the item will be used in the same environment in the military as it was designed for commercially, then the procurement is classified as an NDI category A. Then, the fact that the procurement is NDI category A can be used to determine the phases and milestones that must be used for the program, and this in turn determines approximately how long it will be before the item can be fielded, and so on. By determining if certain criteria are met or not met, and continuing in a forward manner through the reasoning process checking other criteria as appropriate, the information necessary to write the complete paragraph is obtained and/or deduced.

It was also revealed that some overlap exists between the knowledge

needed for the different areas. For example, to tailor the acquisition process (paragraph 3), it must be determined whether or not the NDI strategy can be followed and if so, what NDI category applies. To accurately address the program structure (paragraph 1), the same knowledge is also needed.

Detailed Design

At the completion of the knowledge acquisition phase, the detailed design phase was begun. Information from the problem assessment and knowledge acquisition phases was used to formulate a set of "design considerations" which effectively formed the system design under the rapid prototyping design methodology. The idea of having a separate rule base for each paragraph, as put forth in the conceptual design, was maintained. This idea was extended somewhat by deciding that any particular area of expertise that might be shared between paragraphs (like deciding what NDI category applied) would each be represented in a separate rule base also. The remainder of this section will discuss the design considerations.

The primary consideration in the design of the prototype system turned out to be the preservation of consistency throughout the document. Although this aspect of the system was barely surfaced during the knowledge acquisition phase by alluding to the fact that some paragraphs use the same information, it became a primary driver of the system design. The need for consistency throughout the acquisition strategy had several implications. First, it implied that either an elaborate form of consistency checking with a mechanism for resolution of inconsistencies would have to be developed, or a method for enforcing

consistency (ie., not allowing any inconsistencies to occur in the first place) was needed. The latter was considered the best solution.

Given that inconsistencies would not be allowed to occur, the other implications followed. For example, the user could not be asked for the answer to a particular question more than once for each acquisition strategy. While this was definitely a desired characteristic anyway, it was required to ensure that a user did not answer the question differently for different paragraphs. Since each question was to be asked only once, this implied that some capability was needed to store the answers to the questions and, further, store deductions and decisions derived from those answers to preclude redundant reasoning. Additionally, if the user was ever allowed to change any of his answers to "edit" the acquisition strategy, these new answers would have to be propagated through all the paragraphs developed to that point to ensure that consistency was maintained.

Since an acquisition strategy is a fairly lengthy document, consideration had to be given to the idea that the user might not develop the entire document at one sitting. This fact reinforced the need for a capability to store previous answers, deductions, and decisions so that development of an acquisition strategy could resume. It also implied that a mechanism was needed to allow the user to resume development of the document and that he would probably have to be reminded of which paragraphs had already been developed.

Maintainability was another key consideration during the design of the prototype knowledge-based system. While any good knowledge-based system should embody the concept of performing in the same manner as the expert, this was critical for the prototype. Since all maintenance on

the system would be done by technical personnel in Mr. Causey's office (per the problem assessment), the prototype needed to function logically like an expert to facilitate understanding (and maintenance) by personnel not intimately involved in the development of the system. The system also had to have built-in flexibility to accommodate changes in acquisition policy and guidance. This implied that textual output should be stored separately, when possible, to facilitate maintenance and that the knowledge bases should be designed so that rules would be easily added, changed, or deleted without rewriting the entire knowledge base. (The flexibility consideration is appropriate for any knowledge base and, in fact, flexibility is a major benefit provided by knowledge-based techniques over conventional programming techniques.)

Although the problem assessment phase indicated that a project engineer (the target user) usually was involved with only one acquisition program at a time, consideration was still given to the fact that more than one acquisition strategy might be developed (and, hence, reside) on a single computer. This was especially appropriate assuming that someday the system might evolve to the point that the entire review cycle would be done in a "paperless" mode, thus requiring the reviewers and approvers (at least) to be able to print, view, and possibly edit more than one acquisition strategy per machine. The implication here was that acquisition strategy files would have to be stored, and that the system would have to be able to "recognize" which files applied to which document. Assuming the futuristic paperless review cycle again, a mechanism to save the files to a floppy and to load them to another computer system was also required (if the user was to be shielded from having to know himself which files went with which document).

Tool Selection

Tool selection is one of the most critical decisions to be made during the knowledge-based system development. The selection of an appropriate knowledge-based system development tool can greatly simplify the implementation process, just as "trying to fit the problem to the tool" can complicate implementation.

Several factors influenced which tools were considered for use. First, the information gained during the knowledge acquisition phase indicated that a forward-chaining rule-based system would be appropriate for use in implementing the prototype. This was due to the fact that, to develop a particular paragraph, the expert first ascertained pertinent information about the current status of the problem situation (the program) and used this information to reason forward in an if-then process to arrive at a solution (the completed paragraph). Thus, the reasoning technique is data-driven, and inferential knowledge is most appropriate (as opposed to knowledge about relationships or inheritance).

The second factor that influenced which tools were considered was the need for the prototype system to run on the Z-248 microcomputer. Due to this requirement (and if the concept of supplying many different BELVOIR offices with an "assistant" was to be demonstrated, this was a requirement), only tools capable of operating on a microcomputer were considered. Also, only tools that were available through the Air Force Institute of Technology or the Air Force Wright Aeronautical Laboratories were considered.

Based on the above factors, the following knowledge-based system development tools were considered for the prototype system:

- Insight 2+ (Level Five Research, 1986)
- OPS5 (Brownston, 1985)
- OPS83 (Production System Technologies, 1986)
- Personal Consultant Plus (Texas Instruments, 1986)
- Guru (Micro Data Base Systems, 1986)

Waterman presents six basic questions to ask when choosing a knowledge-based system development tool (Waterman, 1986:Chap 13). These questions were used as a basis for guiding the tool selections process.

The first question to consider is whether the tool provides the developer with the power and sophistication needed to develop the system. All five tools met the criteria for a knowledge engineering language; that is, they all have a pre-determined inference engine (as opposed to a programming language that does not). Any of the tools could have probably been used to build a similar system (with varying degrees of ease), however, their levels of sophistication vary considerable as shown in the discussion of the other questions.

The next question to ask is whether the tool's support facilities are adequate considering the time frame for development. It is here that the features contained in Guru provide a distinct edge over the other tools. The built-in features of Guru include a completely menu-driven system development capability, on-line documentation, easy to use I/O facilities, ability to easily implement interaction with the user (very necessary for this application), comprehensive text processin functions, explanation capability, ability to contro. rule selection criteria, and a database facility for storing knowledge for interaction with one or more rule bases. These features are extremely helpful in a

rapid prototyping environment by allowing the developer to interact with the tool at a higher level and by minimizing support facilities that must be designed and built by the developer. None of the other tools came close to the myriad of support facilities provided by Guru.

Tool reliability and tool maintainability are the third and fourth questions, respectively. Since none of the tools are still under development, nor are any of the tools so old that they are no longer maintained, these two questions did not reveal a particular advantage for any one tool.

The next question to consider is whether the tool contains the features suggested by the needs of the problem. As stated earlier, all the tools meet the primary need for a forward-chaining, rule-based system. Another consideration here is the characteristics of the data. Part of the need for the system is due to the relatively frequent rate at which acquisition regulations and guidance change. Because the knowledge contained in the system will, therefore, require frequent update, the ease by which the rules can be changed is important. As mentioned previously, Guru contains menu-driven facilities to enable even a novice to easily create and edit rules (important since the system will be maintained by personnel in Mr. Causey's office). The rules are also written in an English-like manner, making the content readily understandable. One disadvantage of Guru, however, is that only the first eight characters of rule (and variable) names are used to determine uniqueness. This limits the ability somewhat to have descriptive names (more on this in Chapter VI). OPS5 and OPS83 rules, however, are cryptic and generally unintelligible to anyone except an experienced user of the language and neither contain a menu-driven facility for rule

development or editing. Insight 2+ and Personal Consultant Plus also use rules that are easily understandable and have some type of facility to aid in rule development, but the features are not near as elaborate as those provided by Guru. Another key feature suggested by the needs of the problem is the ability to store data to facilitate document consistency and provide the ability to resume development of a document. While Insight 2+ and Personal Consultant Plus both contain facilities to allow interaction with dBase II and III (and III+ for PC Plus), neither contains a built-in database facility like Guru does. The premise or conclusion of any Guru rule can interact with the data base to modify the data, redefine fields of data, add records or fields, delete records or fields, or simply use the data as "stored knowledge." The interactive data base capabilities provided by Guru, without the necessity of a separate software package or custom interfaces, is a significant advantage for this tool.

The last question, and one of the most important, is whether the tool has the features suggested by the needs of the application. One of these needs was the ability to provide a friendly, simple interface to the user for asking questions, receiving answers, and displaying information. To cause each paragraph of the acquisition strategy to be constructed in a logical, flowing manner within itself, the tool also had to have a mechanism for the control of rule firings. Another need was the ability to have multiple rule bases "open" at one time; that is, allow rules in one rule base to invoke a consultation with another rule base if necessary (for example, if the NDI category was not known, the paragraph 1 rule base could invoke the NDI rule base to figure out what NDI category was appropriate). Only Guru has all of these features.

Guru has the built-in capability for allowing one rule base to call another (up to 50 nested consultations), 50 built-in rule selection strategies, and a well developed user interface capability through the screen forms facility. Another need suggested by the application was the capability to allow the user to input text. Guru has a built-in word processor with an extensive set of features, and provides help screens (available upon user command) to explain what commands are available. In addition, Guru allows for calls to programs outside of Guru which would enable an organization (such as BELVOIR) to standardize on a word processing package, and Guru would be able to interface with it also. The many other features of Guru such as the natural language interface and the ability to handle reasoning with uncertainty, coupled with the capacity for over two million records per data base, also suggested a favorable environment for expansion of the prototype.

Based on consideration of the six questions offered by Waterman as a basis for choosing a knowledge-based system development tool, Guru was chosen as the tool to use for development of the prototype system in this research.

Implementation

As previously alluded to, the implementation of the prototype knowledge-based system allows for development of paragraph 1, Program Structure, and paragraph 3, Tailoring the Acquisition Process, of acquisition strategies for Non-Developmental (NDI) procurements within BELVOIR. The prototype uses three rule bases; a rule base for developing paragraph 1, a rule for developing paragraph 3, and a rule base for determining the appropriate NDI category. The system also utilizes over

50 procedural (Guru "perform" files) and text files, and over 90 screen forms are defined within the system to facilitate interfacing with the user. A database is used to force consistency within the document, to eliminate the need for asking the same question more than once, and to keep the user from having to develop all paragraphs in a single session. Priorities were assigned to each rule to ensure that, within each paragraph, text was written in the correct order.

Figure 9 depicts a sample Guru rule used in the prototype. More detail on the implementation of the paragraph 1 rule base, the paragraph 3 rule base, and the NDI rule base is given in Appendix C, Appendix D, and Appendix E, respectively.

The remainder of this section will discuss the prototype implementation from a functional standpoint. First, the structure and use of the data base will be examined. Then, the eight major functions available to the user from the main menu will be briefly discussed.

Data Base Structure. The design and use of the data base within the prototype system was one of the most critical aspects of the prototype implementation. As mentioned previously, a method had to be developed to enforce consistency throughout an acquisition strategy. In the prototype system, a Guru data base is used for this purpose. Each record in the data base is composed of 17 fields. The first field is used to hold the variable name, and the second holds the variable value. The third and fourth fields hold the reason for the value and the source of the value, respectively. The last 13 fields correspond directly to the paragraphs of the acquisition strategy, and contain an asterisk if the variable is used in that particular paragraph. For example, if the variable name 'CAT' is used to signify what NDI category the procurement

```
RULE:  DECIDEA

       PRIORITY:  85

       IF:        TCAT = "HELP" AND
                  TAVAIL = "yes" AND
                  TINVEN = "yes" AND
                  TENVIRON = "yes"

       THEN:      TCAT = "A"
                  TREASON = "item is a replacement, commercially
                            available, and will be used in the same
                            environment"
                  TSOURCE = "ndixpert, rule DECIDEA"
                  Change DATA.VALUE to "A" for DATA.VAR = "CAT"
                  Change DATA.REASON to TREASON for DATA.VAR = "CAT"
                  Change DATA.SOURCE to TSOURCE for DATA.VAR = "CAT"
                  ndifound = "done"

       NEEDS:     tcat
                  tavail
                  tinven
                  tenviron

       CHANGES:   tcat
                  treason
                  tsource
                  ndifound

       REASON:    Help was asked for in determining the NDI category
                  and user answers indicated category "A."

       COMMENT:   Rule fires if the user asked for help in
                  determining the NDI category and answered that the
                  item is commercially available, the capability is
                  currently in the inventory, and the item will be
                  used in the same environment as commercially.
                  RESULT:  Determines NDI category "A," updates the
                  database, and ends the consultation with the NDI
                  rule base.
```

Figure 9

Sample Guru Rule

is (assume Category A) and this information was decided by the sample rule shown in Figure 9, the record in the data base would appear as follows:

Field	Value
var	CAT
value	A
reason	item is a replacement, commercially available, and will be used in the same environment
source	ndiexpert, rule DECIDEA
P1	*
P2	
P3	*
P4	
P5	
P6	
P7	
P8	
P9	
P10	
P11	
P12	
P13	

This record allows the system to "remember" that the NDI category has already been determined, in this case by the system based on how the user answered questions, and is category A. It also allows the system to remember that the NDI category is important to paragraphs 1 and 3, and that if the NDI category is changed during the edit of either of these paragraphs, the new value must be propagated through the reasoning process of the other.

Each variable in the data base represents a piece of information that is required for the development of paragraph 1 or paragraph 3, or both. Some of the values are determined directly by user answers to questions, some are deduced from user answers, and some can take on default values if the user defers to the system. A total of 47 different variables are used by the prototype to generate paragraphs 1 and 3.

The database records are available to all three rule bases and to all of the procedural files. The use of the data base during an acquisition strategy development session is discussed in more detail when each menu option is examined.

In addition to providing a mechanism for consistency enforcement, the data base had an additional benefit. By using the data base to store all facts needed in the reasoning process, the information used to "solve" the problem is explicitly defined within the system. Having all the necessary information explicitly defined (and thus clearly represented for the prototype developer) made debugging, refining, and enhancing the system during the implementation process much easier. The capability within Guru to examine and print the data base should also have positive effects on system maintenance and future expansion.

Each major function provided to the user on the main menu will now be examined briefly. The eight options provided (in addition to an option to end the session) are as follows:

1. Develop a New Acquisition Strategy
2. Continue Work on an Existing Acquisition Strategy
3. Make Changes to an Existing Acquisition Strategy
4. Print an Acquisition Strategy
5. View an Acquisition Strategy
6. Delete an Acquisition Strategy
7. Save Files for a Program to Diskette
8. Load Files for a Program From Diskette

Figure 10, presented after the discussion of all options, provides a functional system diagram.

Option 1: Develop a New Acquisition Strategy. When this option is selected, the user is prompted for the program name and acronym. The system then calculates a program ID for the program and adds it to a small data base that is used to identify to the prototype system what

programs have acquisition strategies developed or under development. The program ID is also used to identify which files (text files and data base files) go with which acquisition strategy. The system then builds and initializes the data base for the program, and presents the user with a menu of the 13 paragraphs so that the paragraph to be developed can be selected. The paragraphs may be developed in any order.

It is appropriate here to discuss how the system uses the data base to prevent redundant questioning. Paragraphs 1 and 3 both use the NDI category in the textual output and in the reasoning process. Assume the user decides to develop paragraph 1 first; when the system gets to the point that the NDI category is needed it will first check the data base to see if this information is known. If not, the system will invoke a consultation with the NDI rule base to determine the category, and then use the information once it has been provided. When paragraph 3 is developed, the system already knows the NDI category. Even if the user has turned the system off and is developing paragraph 3 a week later, the NDI Category is still stored in the data base. When the NDI category is needed for paragraph 3, the system again checks the data base, this time finding it already knows the information and continues the reasoning process without invoking a consultation with the NDI rule base, thus eliminating redundant questioning. This also eliminates the user's ability to cause the system to have one value for the NDI category in paragraph 1 and a different value in paragraph 3. Although this is a simple concept with a straightforward implementation, it is a key part of the prototype system design and eliminated the need for complicated consistency checking and resolution mechanisms.

Whenever the user is finished developing a paragraph or paragraphs

(he is not forced to do even one), he can select the option to quit and the system returns to the main menu.

Option 2: Continue Work on an Existing Acquisition Strategy. When this option is selected, the system checks the program data base to see what acquisition strategies exist. If none are found, the user is so informed and the main menu is again presented. If one or more acquisition strategies is found, the program names are presented to the user so that he may pick the program to work on. Once the program is chosen, the files for that program are loaded and the user is presented with the paragraph menu as in option 1. All paragraphs already developed, if any, are marked with an asterisk. The user may select any paragraph to work on except for those already developed. When the quit option is selected, the main menu is presented again.

Option 3: Make Changes to an Existing Acquisition Strategy. This option is used by the user to change any answers he previously gave the system while developing an acquisition strategy and have the acquisition strategy "edited" based on the new answers. This option functions exactly like Option 2, except that the only paragraphs the user may pick from the paragraph menu are those that have <u>already</u> been developed. When a paragraph that has already been developed is chosen, the user is presented with a menu of items that may be changed. For example, if paragraph 1 is selected, the following choices are presented:

1. Change NDI Category

Change Answer/Input for:

2. Program Summary
3. Total Contractor Support Management Option
4. Government Furnished Equipment (GFE)
5. Materiel Developer
6. Combat Developer

7. Proponent School
8. Organization with Development Responsibility
9. Organization with Production, Procurement, & Readiness Responsibility
10. Organization with Integration Responsibility
11. Organization with Logistic Responsibility
12. Organization with Evaluation Responsibility
13. Organization with Testing Responsibility
14. Designated DESCOM Depot
15. Quit and Process Changes (if any)

As items are selected to be changed, they are marked with an asterisk. After all items to be changed are selected and marked, the user quits (option 15) and the paragraph (in this case paragraph 1) is re-developed.

During this process of making changes to an existing acquisition strategy, the data base again plays a key part. For each item that was picked by the user, the system goes to the data base and changes the value, reason, and source field of the appropriate variable or variables to blanks. As this is done, the system checks to see what paragraph(s) the variable is used in and if that paragraph has already been developed. When all affected variables are "blanked," the rule base for each affected paragraph is re-consulted (in numerical order). Say, for example the user chose to change the NDI category in paragraph 1, and nothing else. The value, reason, and source for the variable 'NDI' would be blanked in the data base, and the paragraph 1 rule base would be consulted. When the NDI category was needed, the system would check the data base and find that it did not know the category. Thus, the NDI rule base would be consulted just as if the paragraph was being developed initially, and the user would be provided the opportunity to influence a change in the NDI category. When the new category was determined, the paragraph 1 rule base would continue. If all the other

answers previously supplied by the user still applied, then these questions would not be "re-asked" to the user. If additional or different information was needed based on the changes, the appropriate information would be asked or deduced. After paragraph 1 was re-developed, the system would then re-develop paragraph 3 using the new NDI category (assuming paragraph 3 had previously been developed). Again, if all the other information was still correct, the user would not be presented with any questions. When this process was complete, all paragraphs affected by any changes selected for paragraph 1 would now reflect these changes, thereby preserving consistency throughout the document, and the user would have only been required to interact with the system for those items he selected to change. Again, this is a simple concept and easily implemented, but it provides a powerful and very useful capability to the user.

When all requested changes have been processed, the user is returned to the main menu.

Option 4: Print an Acquisition Strategy. When this option is selected, the user is asked to choose which program acquisition strategy is desired for printing. The user is then presented with a menu that provides options for printing any paragraph alone, or for printing the entire document. If a paragraph is selected for printing that has not been developed yet, a checklist for the paragraph is printed. The checklist outlines all the information that is required in the particular paragraph, any applicable references, and possible offices the user could contact for assistance. If the paragraph has been developed, it is, of course, printed. (An option does exist for the printing of only the checklists if the user so desires.)

Another important feature of the prototype system that surfaces here is the justification file. As each paragraph is developed, appropriate text is written to a disk file as information becomes known. At the same time, a justification for the text is written to the justification file for the paragraph. For example, when the user enters the program summary, the statement "The program summary was entered by the user" will be put in the paragraph 1 justification file. If the user defers to the system to discuss NDI in the program summary, the statement "Discussion reference NDI came from the system (user picked this option)" will be put in the paragraph 1 justification file. When paragraph 1 is printed, the justification file for paragraph 1 will be printed on a separate page. In this manner, the user and the maintainer (as well as reviewers if the justifications are attached) can tell what the user was responsible for versus what the system provided. This feature should allow some of the reasoning process to be apparent to the user, and contribute significantly to rule base maintenance, correctness, and troubleshooting.

After any printing that is desired has been completed, the user is returned to the main menu.

Option 5: View an Acquisition Strategy. This option functions exactly like option 4, except that output is directed to the screen instead of the printer. Again if a paragraph is selected for viewing that has not yet been developed, the paragraph checklist is presented.

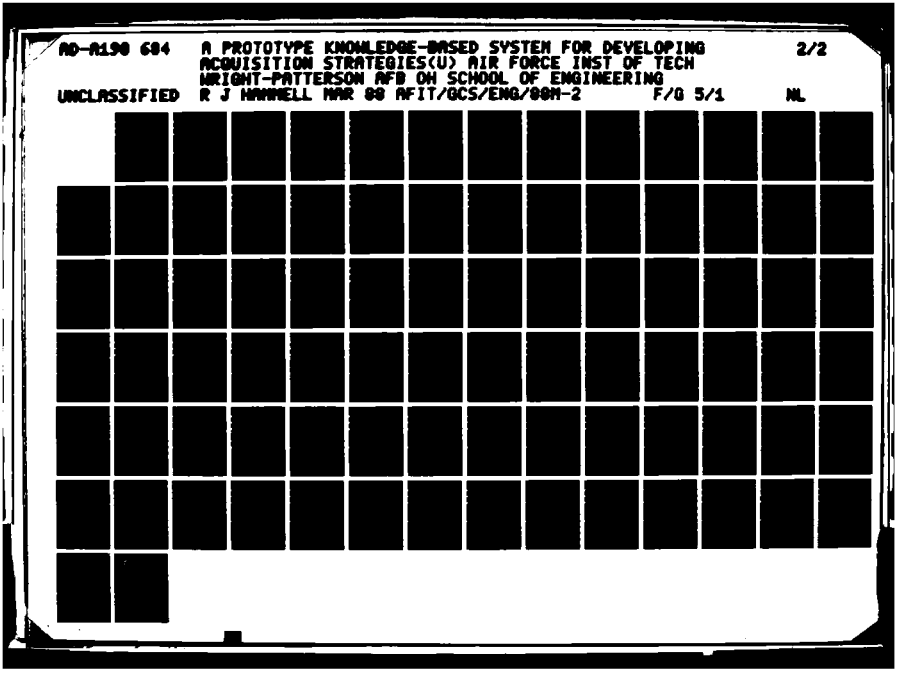
Option 6: Delete an Acquisition Strategy. When this option is selected, the user is presented with the program names of all existing acquisition strategies to choose which one is to be deleted. After a selection is made, the user is asked to confirm or abort the deletion.

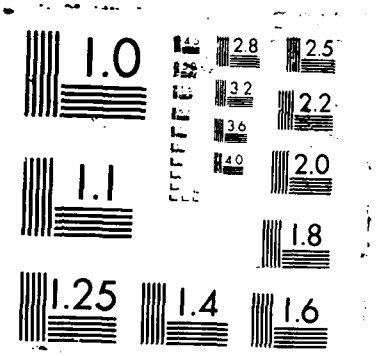
If aborted, the user is simply returned to the main menu. If the deletion is confirmed, the acquisition strategy files for the selected program are deleted, and the program ID is removed from the program data base that is used by the system to track existing acquisition strategies. The user is then returned to the main menu.

Option 7: Save Files for a Program to Diskette. This option enables the user to copy acquisition strategy files for the chosen program to a floppy diskette. Only files applicable to the chosen acquisition strategy are copied to the diskette, and the program ID and other information used to identify the program is also copied to the diskette. If files for the chosen acquisition strategy already exist on the diskette, the user is given the option to overwrite the files or to abort the save process. The Save Files option can be used to backup acquisition strategy files, but must be used if the acquisition strategy is to be transported to another microcomputer.

Option 8: Load Files for a Program to Diskette. This option is used to load files from a diskette that were previously saved using Option 7. When the Load Files option is chosen, the system checks the diskette to locate the program ID information for the chosen acquisition strategy, and adds this information to the system program data base. The acquisition strategy files are then copied from the floppy onto the hard disk, and this program will appear on the program selection menu until it is deleted. If the selected acquisition strategy already exists within the system, the user is given the option to overwrite the files or abort the load process. The user is returned to the main menu upon completion of the load process or its abortion.

Option 0: Examine a Program Trace. Although this option does n


AD-A190 684
A PROTOTYPE KNOWLEDGE-BASED SYSTEM FOR DEVELOPING ACQUISITION STRATEGIES(U) AIR FORCE INST OF TECH WRIGHT-PATTERSON AFB OH SCHOOL OF ENGINEERING
2/2
UNCLASSIFIED
R J HAMMELL MAR 88 AFIT/GCS/ENG/88M-2
F/G 5/1
NL


1.0
1.1
1.25
1.4
1.6
1.8
2.0
2.2
2.5
2.8
3.2
3.6
4.0

appear on the menu, the system has the capability to show a trace of all rules fired from a previous consultation. This capability was primarily developed to provide insight into the performance of the rule base for the developer, but was left for use by the system maintainer if desired. When a program acquisition strategy is selected for development, continued work, or changes, every rule within any of the rule bases adds an entry into the trace data base. Each entry tells which rule was fired, which rule base the rule is in, and why the rule was fired. As long as work continues on the same program acquisition strategy, even if the system has been exited or the computer powered off, the rule traces will continue to be added to the trace data base. As soon as work starts on a different program, however, the data base is cleared and the trace entries will begin to accumulate all over again.

The trace capability is limited to only the last (or current) program acquisition strategy on which work was performed. The rule traces may be viewed or printed separately by rule base, or together in the order in which the rules were fired. The main menu is presented when the option is exited.

As mentioned at the beginning of this section, the prototype system only provides the capability to develop paragraphs 1 and 3 at this point. It is important to note, however, that all functions, files, and modules were implemented with all of the code necessary to support all 13 paragraphs. That is, menus that offer paragraphs for development, printing, and viewing include all paragraphs, and stubs are in place that identify these capabilities as "Under Development" if selected. The framework to support implementation of all 13 paragraphs is in place. To implement any of the other paragraphs, only the following is

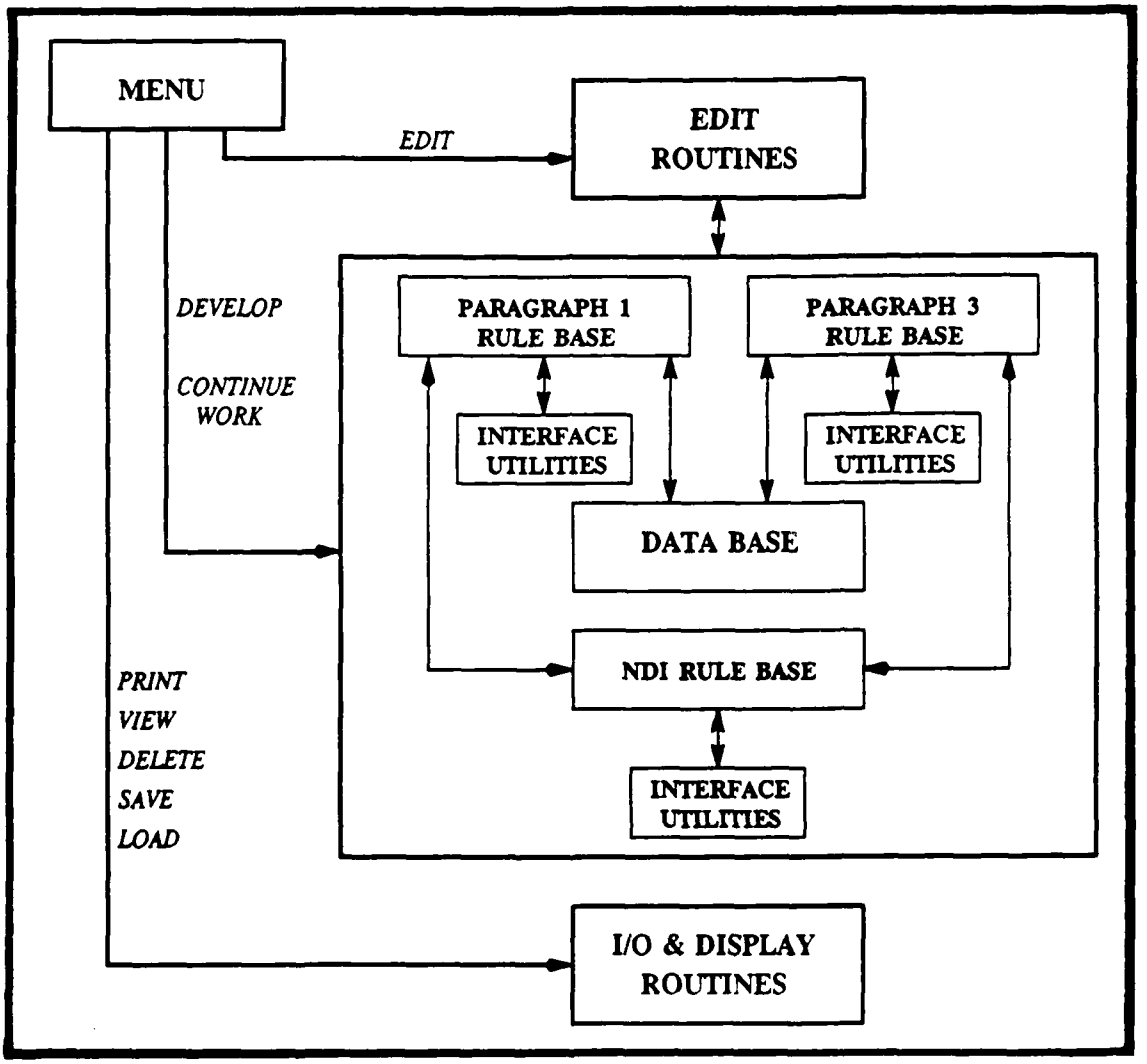


Figure 10

Prototype System Functional Diagram

required: (1) develop the rule base for the paragraph (this is non-trivial), (2) add the new variables used in the new rule base to the code that generates the acquisition strategy data base, (3) add code to offer appropriate variables for changes within menu option 3, and (4) remove the stubs for the new paragraph.

The entire framework for all paragraphs was designed and implemented for two reasons. First, the author felt it was required to adequately demonstrate the concept of applying a knowledge-based system on a microcomputer to the problem of generating an acquisition strategy. Secondly, and more importantly, the existence of such a framework will hopefully stimulate BELVOIR to continue expansion of the prototype, and even allow other organizations to add and/or change knowledge to permit them to use the prototype.

Summary

In this chapter, phases in the development of a knowledge-based system and the approach used in this research were discussed. This was followed by a presentation of the pre-prototype development, the knowledge acquisition phase, the prototype system design, and the tool selection process. Finally, the prototype system implementation was examined. The methodology and efforts detailed led to a functional prototype knowledge-based system that assists the user in developing paragraphs 1 and 3 of an acquisition strategy for NDI procurements. In addition, the framework is implemented to support the development of all the acquisition strategy paragraphs.

In the next chapter, the evaluation of the functional prototype is presented.

V. EVALUATION

In this chapter, the evaluation of the prototype system is presented. The evaluation was structured as outlined in Chapter I, and the material presented here is organized in the same manner. First, system performance is examined, followed by sections discussing expandability of the system and maintainability of the system. Finally, applicability of the knowledge-based system approach to the problem domain of developing acquisition strategies is examined.

System Performance

To address system performance, the following aspects of the prototype system were considered:

- "correctness" of the knowledge,
- "correctness" of the lines of reasoning,
- "correctness" and "completeness" of the text written by the system,
- the user interface, and
- verification of the code.

For the first four aspects listed above, the BELVOIR experts that provided information during the knowledge acquisition phase were asked to provide the evaluation. The author provided the evaluation of the fifth system aspect.

When the implementation of the prototype system capabilities was approximately 85% complete (all the "knowledge" was implemented, however), the BELVOIR experts were asked to provide a preliminary, informal evaluation of the prototype. To accomplish this, Mr. Dan Causey (paragraph 1) and Mr. Brian David (paragraph 3) came to Wright Patterson

AFB. The evaluation consisted of three parts: first, the operation of the (incomplete) prototype was demonstrated to the experts. Then, each rule in the individual rule bases was explained to the experts and validated (or changed). Finally, the textual output generated by the system was examined, both as individual sentences or groups of sentences, and in the context of the reasoning process that caused the text to be produced. This three step process ensured that the user interface, the correctness of the knowledge, the correctness of the lines of reasoning, and the correctness and completeness of the written text were all addressed by the experts.

Based on the preliminary evaluation, several changes and additions were made to the prototype. None of the knowledge in the system was considered "incorrect" by the experts. However, suggestions were made to "fine tune" the way in which some of the information was gathered from the user, and some of the textual output was revised. Also, suggestions were made to get additional information from the user so the textual output would be more complete. Thus, the reasoning could be judged to be "incorrect" in the sense that it gave a useful but incomplete result.

An example of the type of changes suggested to "fine tune" the system can be illustrated by the way in which information about the specific agencies involved in the procurement was obtained. Initially, the user was asked, one by one, which agency was the materiel developer, the combat developer, and so on, for twelve different program responsibilities. During the evaluation, it was determined by Mr. Causey that there were actually three "fields of endeavor" that covered most procurements within BELVOIR, and each field had a fairly standard set of

"default" agencies for the different responsibilities. The suggestion was made (and subsequently implemented) to allow the user to select a field of endeavor and, based on the one picked, present a "default" list of organizations matched up with the various responsibilities. By allowing the user to accept the defaults that were correct, the only questions about agency responsibilities to be asked would be the ones for which the default agencies had been rejected.

An example of an addition that was made to make the textual output more complete was the addition of program milestones to paragraph 3. After examining an example paragraph 3 developed by the prototype, the experts suggested that a few basic program milestones should also be discussed in this paragraph. A methodology for prompting the user for milestone completion dates, and "knowledge" for predicting completion dates of those not yet completed, was provided by the experts and this capability was subsequently added.

The changes and additions recommended during the preliminary evaluation can be primarily attributed to two factors. First, the experts "learned" more about developing the paragraphs. That is, their understanding of the task of developing an acquisition strategy paragraph and their reasoning during the development became more apparent to them as a result of participating in the knowledge acquisition and evaluation process. This is illustrated by the realization that three primary fields of endeavor are used to suggest which agencies are involved for a particular program. Secondly, the concept the experts had of the potential capability of the system changed during the preliminary evaluation. This is shown by the addition of the program milestones. It seemed as though the omission of this aspect initially was not really

due to error, but due more to lack of confidence that the prototype would be able to handle everything that was really needed for the paragraph. This is reinforced by the fact that many additions were suggested as a result of the preliminary evaluation; the suggested additions increased the number of rules by approximately 25%.

After the preliminary evaluation of the prototype, the additions and changes were incorporated into the system and the remainder of the prototype was developed. Then, a copy of the prototype system software, along with instructions for its use was mailed to Mr. Causey for a formal evaluation. (Due to time constraints and availability problems, Mr. Causey was unable to get Mr. David to evaluate the prototype also, which was the original intent. Mr. Causey served to evaluate the entire system.) He was asked to use the following process: Take two existing acquisition strategies and pretend that paragraphs 1 and 3 were not developed. Then, use the system to develop paragraphs 1 and 3 and for each acquisition strategy address the following:

- Are the paragraphs generated by the system identical to those written by people?
  - -- If so, is that good?
  - -- If not, are the ones developed by the system "good" or "bad" different, or different but equal (and why)?
- Is the "knowledge" in the system correct and if not, what must be changed?
- Is the line of reasoning exhibited by the system correct and if not, what must be changed?
- Is the textual output generated by the system complete and correct given the acquisition program parameters and if not, what must be changed and/or added?
- What changes in the user interface are needed?

The evaluation provided by Mr. Causey (Causey, 1987) was very positive. The knowledge contained in the system was considered to be

"correct" and provided "a clear line of reasoning from the questions to the text actually generated in the paragraph." While the evaluation stated that the paragraphs generated by the system were not identical to those developed by people, "the meaning of the paragraphs is consistent." The textual output was judged "more than adequate" and "in some cases" Mr. Causey preferred the "system text to that generated by the original author." The evaluation also stated that the text provided by the system was more consistent across both acquisition strategies, which was considered a positive benefit "for the sake of producing clear, standard acquisition strategies." The evaluation included a comment alluding to the fact that some of the aspects addressed by the system in the text may need to be tailored for the specific situation, but also provided that it was "best to assure coverage of all standard items in any acquisition strategy."

Comments addressing the user interface were all positive. It was termed "very nice," and "robust" in that "an occasional typing error" was tolerated very well. Mr. Causey also stated that the "use of colors is very helpful and the questions 'feel' like what a friendly expert would ask." Although no comment was made in the written evaluation, previous verbal comments from Mr. Causey indicated that the capability to edit the paragraphs after development was very useful. The fact that consistency was enforced during the edit of any paragraph, as well as during initial paragraph development, had also been termed extremely beneficial.

No required changes or additions were identified in the evaluation. Based on the evaluation from Mr. Causey, the first four aspects of system performance were considered satisfactory. The sample output from

one of the prototype-generated documents produced during the evaluation by Mr. Causey is included at Appendix F. This output is for the LAMP-H program and the original (complete) acquisition strategy for this program is shown at Appendix B (it was used as the problem assessment example).

The last aspect of system performance to be examined is that of verification of the code. Verification is the process of ensuring that the code is free of bugs (Margot, 1987). Since this is a prototype system, no attempt was made to develop a "proof" that the the software is entirely free of bugs. The prototype was developed in a modular fashion, which allowed for individual testing of each "module" in most cases. In the instances where individual testing of the "new" code was not possible, it was possible to ensure that the new code was tested in conjunction with only previously tested code. Every rule and every decision branch in the system was used at least once, and in most cases many times. Due to the many combinations possible in a system of this size, especially with the added complexity of the reasoning that takes place over the knowledge bases, it was not possible to test for every conceivable situation. Extensive testing was done on the system to try and ensure that unknown errors would not be critical ones. However, since testing cannot prove the absence of bugs but only their existence, the code was not verified in the strict sense of the word.

Expandability

An opportunity to judge the expandability of the system was presented after the preliminary evaluation by the BELVOIR experts. As previously stated, several additions were suggested and the total number

of rules in the system was increased by approximately 25%. Because a separate knowledge base was used for each "domain" of knowledge, the addition of rules was very simple. Further, the use of the rule priority mechanisms provided by Guru allowed the addition of the rules anywhere within the knowledge base while still ensuring that the rules would be considered in the proper order (to allow structured text generation).

Additions other than rules were also easily accomplished. The capability to have many "perform" files in Guru (procedural code used somewhat like sub-programs), made it simple to increase capability of existing files and develop new files as needed. This "modularity" minimized the possibility of inducing errors into the system and facilitated testing of the additions.

The other aspect of expandability is that of being able to extend the prototype to develop all paragraphs in an acquisition strategy. As stated in Chapter IV, the entire framework to support all functions for all paragraphs was built-in to the prototype. Only the knowledge needed to develop the other paragraphs, and functions peculiar to that knowledge, must be added. It is sincerely hoped that this will encourage further expansion and use of the prototype.

Maintainability

The maintainability aspect of the prototype system has not actually been able to be evaluated to this point. However, the features that allowed for the ease of expansion should also provide for ease of maintenance. The multiple knowledge bases and multiple perform files, all of which are logically separated, should support revisions to rules,

user interface mechanisms, wording of questions, and changes in textual output very easily. The justification file printed with each paragraph, along with the detailed information provided through the trace feature, should also facilitate maintenance of the system if incorrect knowledge or lines of reasoning are suspected at some point. The system was designed to follow a reasoning process that could be easily understood, and this in itself should make any necessary debugging, changing, or updating less complicated. The eight character limit on Guru rule and variable names hinders the ability to provide descriptive, self-documenting names. However, the "reason" and "comment" fields provided by Guru for use within each rule can be used to overcome this disadvantage to some extent.

As stated in the tool selection portion of Chapter IV, the many features provided by Guru should also facilitate the maintenance process. By providing a user-friendly environment with the capability to perform all development functions through menus, only minimal knowledge of Guru is necessary for even a novice to use the tool. Since the system will be maintained by personnel not involved in the details of the system development (and, thus, not intimately familiar with Guru), the ease of use of Guru should be an added benefit.

Applicability of the Knowledge-Based Approach

The development of the prototype system has clearly demonstrated the applicability of the knowledge-based approach to the problem domain of developing acquisition strategies. The benefits of ease of expansion and maintainability that are provided by knowledge-based techniques are in themselves enough to demonstrate the applicability of the approach.

This is due to the fact that, as stated in the problem assessment, one of the biggest factors driving the need for the system is the frequent changes in acquisition guidance and policy. With a "conventional" program, a change in acquisition procedures could very easily cause significant amounts of the program to be rewritten or restructured, while the knowledge-based system could perhaps handle the same change with only a modification or addition of a few rules.

The prototype demonstrated that the concept of integrating the expertise of several domain experts to assist the project engineer is a viable one. As illustrated by the problem assessment, the primary obstacle faced by every project engineer confronted with the task of developing an acquisition strategy was the need for knowledge in many different areas. By providing him with an "assistant" that has access to the required knowledge, the capabilities and productivity of the project engineer is immediately increased. An added benefit is that the workload on the "experts" is immediately decreased, thereby boosting their productivity.

The time required to generate an initial draft of paragraphs 1 and 3, especially for a novice, is greatly decreased. Using the prototype system, both paragraphs could probably be developed in less than two hours, including typing time for the program summary and other text the user may have to enter. This is compared to the more than two weeks that was estimated (during the problem assessment phase) as necessary for a novice to develop these paragraphs (see Table II). The dramatic decrease in time can be attributed to the fact that a user of the prototype does not have to search regulations or seek out experienced project engineers to know what must be addressed in the paragraphs. The system

presents the required elements to the user and "assists" with some knowledge to help address them. If the user has to look up some information about the program, such as details for the program summary or organizations that are responsible for various program aspects, the time could be longer than two hours. In any case, the time required to develop paragraphs 1 and 3 with the prototype should be much shorter, and all standard items in the paragraphs will be covered.

In addition, the ability of a knowledge-based system to simulate a reasoning process that can be easily understood by humans is a benefit. This capability not only provides a way for the system to "explain itself," but also facilitates user acceptance by being a system that can be understood. If a system is used more, the benefits provided by the system will be exploited and if it is understood by the user, the chances are greater that the user can suggest improvements (thus increasing the benefits of the system).

The whole methodology associated with the development of a knowledge-based system also demonstrates the applicability of the approach. It would be virtually impossible for an expert or group of experts to provide specifications detailed enough to permit the design of a conventional program to generate acquisition strategies using the traditional 'waterfall' approach. However, as illustrated by the positive prototype evaluation, a developer relatively unfamiliar with the problem domain can produce a useful and "correct" system in a relatively short period of time by using the rapid prototyping methodology, simple knowledge acquisition techniques, and by keeping the experts involved in identifying the "specifications" of the system. Knowledge-based system techniques support this approach, and knowledge-based system development

tools provide and environment to maximize and exploit the benefits of the approach.

Table III presents a summary of the prototype evaluation efforts.

TABLE III

Prototype Evaluation Summary

PROTOTYPE ASPECT	PRE-EVALUATION	FORMAL EVALUATION	SUBJECTIVE EVALUATION
"Knowledge"	*	*	
Line of Reasoning	*	*	
Text Completeness	*	*	
Text Correctness	*	*	
User Interface	*	*	
Code	(Tested Throughout Implementation)		
System Expandability	*		*
System Maintainability			*
Applicability of Approach			*

RESULTS:
- Changes and additions to the prototype were made based on the preliminary evaluation.
- Formal evaluation was very positive. No changes to prototype were suggested.
- Positive subjective evaluation for system expandability and maintainability, and for applicability of the knowledge-based system approach.

NOTE :
- Pre-evaluation performed by BELVOIR experts and author.
- Formal evaluation performed by BELVOIR expert (Mr. Causey).
- Subjective evaluation done by author.

Summary

In this chapter, the evaluation of the prototype knowledge-based system was presented. First, the system performance aspect of the prototype was examined. Then, the expandability of the system and maintainability of the system were discussed. Lastly, applicability of the knowledge-based system approach to the problem domain of developing acquisition strategies was presented.

The evaluation of the prototype system from Mr. Causey was very positive. The expandability and maintainability aspects of the prototype and, in general, any knowledge-based system, illustrated that the frequent updates and changes in the acquisition environment can be accommodated. Additionally, the prototype validated that concept of combining the expertise and experience of many experts into an "assistant" to help the project engineer develop an acquisition strategy. Overall, the prototype knowledge-based system designed and implemented in this research has clearly demonstrated the applicability of the knowledge-based system approach to the problem domain of developing acquisition strategies.

VI. CONCLUSIONS AND RECOMMENDATIONS

Conclusions

The prototype knowledge-based system developed in this research demonstrated the effectiveness of applying knowledge-based system techniques to the problem of developing acquisition strategies. By utilizing knowledge-based system development techniques and a knowledge-based system development tool, a useful prototype was developed in a relatively short period of time to help solve a problem that defies easy definition.

The prototype system also provides a foundation for the contracting of a complete system to assist in developing the entire acquisition strategy document. The system will not only serve to provide the contractor with a foundation of knowledge and a framework of features, but can also be used as a valuable demonstration aid within BELVOIR to educate potential users and experts. The expected benefits from the prototype in this area were explicitly addressed in the system evaluation done by Mr. Causey:

> I plan to use this expert system as a clear example of what this kind of computer program can do for BELVOIR and as a selling point to get support for completing the work on the expert system for acquisition strategies being done by DSMC for BELVOIR.
>
> (Causey, 1987)

The effort required during this research has illustrated (at least to the author) that the actual coding is only a small part of the effort required to develop a knowledge-based system. This is not to say that the time required to implement and test the system was trivial. This process was, indeed, very time consuming. However, the critical part of the entire project was the effort made to get the "right" knowledge. In

fact, getting the necessary knowledge was interrelated with the process of developing the system (as it usually is when a rapid prototyping methodology is used). That is, getting the "right" knowledge was dependent upon experimenting with the system itself. This was illustrated by the changes and additions to the prototype suggested during the preliminary evaluation. By using a good knowledge-based system development tool, the coding aspect of the development process was simplified. The process of interacting with several other humans, and trying to get not only the necessary knowledge but also the reasoning processes used by the experts, is a very complex task. The requirement to represent the knowledge and reasoning in a computer further complicates the effort.

While the process of knowledge acquisition is complex, this research also proves it is possible to do well. The approach of developing a pre-prototype for demonstration to the experts, following general knowledge acquisition guidelines, and keeping the experts involved in the system development through periodic evaluations enabled the development of a prototype that required very little change at each step. Indeed, the author is certain that the extremely positive system evaluation can be directly attributed to the knowledge acquisition methodology used and the use of a preliminary evaluation of the incomplete prototype.

The concept of using a data base to provide simple but effective enforcement of consistency throughout a document was also illustrated by the prototype system. As discussed in previous chapters, consistency among the paragraphs of an acquisition strategy is absolutely essential. The use of a data base, to which the knowledge-based system has access, proved to be an excellent method of providing a mechanism to enforce the

required consistency. The use of the data base also served to explicitly define within the system all facts needed throughout the reasoning process. Additionally, the data base design process forced the system developer to specifically address the facts that were being used, thus providing a more structured understanding of the information required, reasoning used, and knowledge represented within the system.

Recommendations

Development Tool Considerations. While Guru provides an excellent environment for knowledge-based system development, it does have a few drawbacks that should be considered if the tool is to be used for developing large-scale microcomputer-based systems. One drawback is the length limitation on Guru rule names and variable names. Although Guru will allow long rule and variable names, it only uses the first eight characters in the names and does not allow the use of the underscore character in the names. In large-scale system development it would be extremely difficult for the developer to keep track of the first eight characters of every rule and variable name to ensure that each was unique, so the limitation is effectively eight characters. While this did not seem to be much of a drawback initially, as the prototype got more complex the eight character limit became a definite hindrance. It does not lend very much to code readability, either.

Another drawback to Guru is the lack of a built-in function to automatically analyze the rules for potential problems. While Guru does compile the rule bases and perform syntax checking, it does not check for rule inconsistencies (redundant rules, conflicting rules, subsumed rules, unnecessary "if" conditions, circular-rule chains) or rule incom-

pleteness (unreferenced attribute values, dead-end goals, unreachable conclusions, dead-end "if" conditions) (Nguyen and others, 1987). This was not a significant problem in the development of the prototype system due to the relatively small number of rules, but the ability to check rule consistency and completeness in a very large system would provide an invaluable benefit. One could argue that this capability is a must to ensure correct and efficient knowledge bases.

The greatest problem that was encountered while using Guru was in the area of memory management. The development of the prototype was done on a Zenith-248 microcomputer with 640 kilobytes of memory, which meets the stated requirements for Guru software. Occasionally, when the system was run after extensive text editing or after performing several DOS commands, Guru would abort system operation, issue the message "not enough room in memory," and return to the DOS environment. On other occasions, Guru would continue system operation but DOS commands embedded within the system would not be processed due to a lack of memory, and no indication that this was happening would be provided. (There is a 'reserve()' command that reserves the amount of memory specified by the parameter for use by external programs, but this just transfers the problem back to a lack of memory for Guru itself.) In fairness, however, when the prototype was run on a "fresh" system (boot the computer, go straight to Guru and do nothing else before using the prototype) no memory problems were ever encountered even after multiple paragraphs and multiple acquisition strategies were developed. The memory problems were encountered in Guru Versions 1.0 and 1.1 (the latest release). Both of these versions contain a 'release' command that is supposed to free memory in use by previous Guru commands, but

testing with the DOS chkdsk command did not give any conclusive results that the release command did any good.

While the memory problems do not cause disaster when encountered during use of the prototype (a recovery routine hado be written to restart from a known state), they could prove to be a serious irritant for users, especially if the microcomputer is used for many other purposes and memory resident programs are loaded during normal system boot. Before Guru is selected for use in developing a knowledge-based system with more than a hundred or so rules (or a significant amount of arrays), further investigation into the memory management problems is highly recommended.

Knowledge Acquisition. The knowledge acquisition methodology used in this research proved to be highly effective, though the approach was certainly not the only one that could have produced this result. However, the development of the pre-prototype system for use during the knowledge acquisition phase is considered (by the author) to be a significant contributor to the success of the knowledge acquisition process. Development of the pre-prototype served not only to provide a demonstration system which helped spark user and expert interest, but also forced the prototype system developer to consider rule design and rule interaction as related to the specific problem domain. Additionally, development of the pre-prototype provided a useful mechanism to acquaint the developer with written acquisition policy and guidance, thereby providing some "training" in the problem domain. Just as development of a prototype will help ensure success of the final system, development of a pre-prototype will greatly increase the accuracy of the prototype, as well as assist in climbing the learning curve.

Additional Research. The features provided by the prototype system developed in this thesis effort not only furnish a functional demonstration of an application of knowledge-based system techniques, but actually provide a development tool for developing systems to generate acquisition strategies. Since the prototype provides the entire framework for all acquisition strategy paragraphs, adding, changing, and/or deleting the knowledge contained in the system is basically all that is necessary to develop a system to generate acquisition strategies tailored to any branch of the military, or to any specific organization's needs. Adding the required knowledge (and the required variables and any new user interfaces), however, still requires some familiarity with the system design and with Guru. Enhancements could be made to the prototype to facilitate the process and, perhaps, demonstrate benefits that could be transferred to the final system.

For example, several possible extensions could be to refine the prototype framework to provide an automated capability to: add and edit rules; add new variables to the data base; generate required "screens" to prompt the user; and present the user with the allowed edit choices within each paragraph. Providing an automated capability to generate rules for Guru would be a substantial undertaking, but the other extensions could be implemented fairly easily by redesigning the data base already used for each acquisition program. The addition of any or all of the above capabilities would certainly enhance the maintainability and expandability of the prototype system, and would probably do the same for any knowledge-based system.

In developing a complete system to generate acquisition strategies, the potential benefits of several ideas not surfaced as requirements in

Chapter III should be considered. For example, thought should be given to designing the system to facilitate a "paperless" review cycle. Under this concept, the document would be circulated via floppy disks (or better yet, made available to reviewers via the INFOMAIL network in place within BELVOIR). The reviewer could make suggested changes to the document (which the system would need to be able to flag by some means so they could be readily identified) and then send the "corrected" version back. It would also be beneficial if the capability existed for the system to "read" and consolidate all the recommended changes from the copies that came back, without losing the capability to identify which reviewer made which change.

Along these same lines, perhaps a mechanism could be developed to historically record changes made to a document throughout the program life, as well as capture the rationale for those changes. To the author's knowledge, no history is kept of changes, and the reason for the changes, made to an acquisition strategy as it is updated from life cycle phase to life cycle phase. The ability to do this suggests the potential for a knowledge-based system to actually develop expertise of its own in addition to the expertise explicitly coded in the knowledge bases. Additionally, the capability to query this information could provide an acquisition developer, reviewer, or approver with valuable insight as to why the document contains the information it does. Carrying these ideas one step further, the system could have the capability to "replay" all the knowledge it has about a program, thus adding a new dimension to the knowledge-based system "explanation" feature. The combination of all these capabilities could go a long way toward developing an automated training program for new or perspective project

engineers, which does not exist within BELVOIR today. They would also contribute a great deal to maintenance of the knowledge base.

While some of the above features might require "pushing the state-of-the-art" somewhat, they would be very useful and are appropriate for research. Another concept in this category is reasoning by analogy (Barr and others, 1981:146, 158, 177, 200-206). If a knowledge-based system for assisting in the development of acquisition strategies was capable of analogical reasoning, the system could, perhaps, be able to suggest acquisition strategies from similar programs to use as examples (after details of the new program were given to it). The system might even be able to retrieve appropriate text from the similar program documents itself, and present the text to the user for acceptance or rejection. If the capability for reasoning by analogy was coupled with a historical record of the development of previous acquisition strategies, the system might even be able to tailor questions presented to the user based on which questions proved "important" during the development of acquisition strategies for the similar programs.

Another area of current research that appears to present enormous potential for application to document development is that of hypertext. As stated in Chapter II, hypertext provides the ability to manipulate ideas as objects, and to link text or pieces of text together to form a document. This capability could provide tremendous assistance in the creative task of developing an acquisition strategy or other program management document. The Intelligent Systems Branch of the Air Force Human Resources Lab (HRL) at Brooks AFB, Texas, is currently conducting research into military applications for hypertext. One application being investigated is applying hypertext to the task of developing

Statements of Work. A conversation with members of the HRL indicated an interest in exploring the integration of hypertext with the prototype system developed in this research. One possible scenario (of many) would be to allow the knowledge-based system to gather information about an acquisition program, and then turn control over to the appropriate hypertext knowledge base to provide applicable pieces of text that could be manipulated to form the acquisition strategy. The author sincerely hopes that future research will be done in this area, since the potential benefit is enormous.

Summary

This thesis effort has successfully accomplished the primary goal of demonstrating the effectiveness of applying knowledge-based system techniques to the problem of developing an acquisition strategy. The secondary goals of providing a basis for contracting a complete system, providing a framework of features and knowledge from which the contractor can work, and furnishing a functional demonstration to help educate people and gather support for the complete system within BELVOIR have also been accomplished. Continued efforts in the area of knowledge-based system "assistants" planned by the Defense Systems Management College has the potential to provide enormous benefits to project engineers that must develop acquisition strategies. As knowledge-based systems continue to be applied more and more to management applications, the additional potential exists to provide the same benefits to all areas of program management. This thesis will, hopefully, provide a stepping-stone to that end.

Appendix A

PROBLEM ASSESSMENT CHECKLIST

1. Problem Definition

- Stated as concisely as possible, what is the present problem with developing Acquisition Strategies that is to be solved? (Care must be given to distinguish the problem(s) from the symptoms.)

- How does the above problem impact the organization? (E.g., wasted resources (time, money, effort), adverse impact on programs, delay in getting equipment to the field, inconsistent documents from same organization, etc.)

- What is the value-added that any solution must have? (E.g., should the solution: save money, increase efficiency, increase quality and/or consistency of documents, etc.)

2. Current Practice

- What current systems and procedures, computerized or not, are used to produce Acquisition Strategies today?

- Is expertise critical to the ability to develop an Acquisition Strategy? (E.g., is specialized knowledge required; that is, something more than common sense.)

- If so, where is the expertise? In the heads of people? If so, who are they? In regulations, policies, or manuals?

- How long does it take the most skilled people (the experts) to produce a typical Acquisition Strategy?

- How long does it take an average-performance employee to produce the same document?

- Are there training programs? If so, how effective are they?

- What is the criteria for "failure"? How does it occur?

- What are the error rates and costs associated with not producing an Acquisition Strategy correctly?

- Would an automated knowledge system have to develop an Acquisition Strategy in the same way as it is done now?

- How does information flow within the organization to either help or hinder the development of an Acquisition Strategy?

3. Characterization of the Problem that the Expert is Solving

-The nature of the solution:

- Satisfice or optimize? (I.e., is it sufficient to find one satisfactory solution, or must one generate, potentially, many possible solutions and then select from the optimal one, or perhaps even find and report all solutions?)

-Is the task of developing an Acquisition Strategy one of selecting from a set of pre-defined solutions, or a plan for constructing something that cannot be defined in advance?

- How many potential solutions are there to a typical problem (tens, hundreds)?

- Is the set of all possible solutions different from the set of solutions that the expert uses? (I.e., does the expert immediately cast out a very large fraction of possible solutions and focus attention on a small but highly probable subset?)

- The nature of the input data:

- How many items (tens, hundreds)?

- How correct is the data (that is, does incorrect information ever produce false starts, causing an Acquisition Strategy to be redone)? How often does this occur?

- Is the data static or dynamic? (E.g., does it ever change while the Acquisition Strategy is being developed? If so, does it impact the analysis?)

- The nature of the reasoning:

- What kinds of reasoning have to be done? (E.g., can one make conclusions with certainty, given the truth of particular facts, or is reasoning under conditions of uncertainty required? Is it necessary to make guesses? Is it necessary to search through many possible hypotheses in order to arrive at a solution?

- What kinds of strategies are used to control the reasoning? (E.g., does the expert always gather certain initial data/evidence before starting to develop an Acquisition Strategy?)

- To help answer the above questions, it is suggested that at least one example case be examined in as much detail as possible. The case should be non-trivial, representative, and well understood.

4. Characterization of a Successful Knowledge System

- If a knowledge system were developed and used with the organization, what features must it have? (E.g., graphics, integration with current software or databases, very fast response, zero incorrect solutions?)

- What does the organization perceive to be the benefits in having a knowledge system to assist is developing Acquisition Strategies? (E.g., preserving the expertise of scarce or retiring personnel, improving the productivity of average workers, training new employees, better quality documents, more consistency, quicker development?)

- What are the direct impacts to the organization of developing a knowledge system to assist in developing Acquisition Strategies? (E.g., if your best expert is temporarily assigned to the project, how will it impact current operations?)

- What are the pitfalls that a solution must avoid? (E.g., avoid technical jargon in asking questions or explaining answers, assume the user cannot type, etc.?)

- How easily would the knowledge system be integrated within the organization?

5. Defining a Prototype

- What would a prototype (i.e., a system that could be designed and developed quickly) need to do? (E.g., should the prototype be a convincing demonstration that a knowledge system is appropriate for assisting in the development of Acquisition Strategies, or should it be a core system that exhibits expert performance on a small but important piece of the whole problem?)

- What immediate problem will the prototype solve?

- What are the different bodies of knowledge that are needed?

- Does the prototype have to be integrated with any existing system(s)?

- How will it convince the organization that a knowledge system is appropriate for developing Acquisition Strategies? Who has to be convinced?

- What characterizes a successful prototype? (i.e., what are the important acceptance criteria?)

6. Evolution of the Prototype

- How can the prototype be expanded into a complete system?

- How is the environment in which the complete system will reside different from the environment for the prototype?

- How are the hardware requirements for the complete system different from the hardware on which the prototype is developed?

- What is the possible obsolescence of the complete system?

- What will be its long-term maintenance and enhancement requirements?

7. Resources Required

- Data: How many different typical cases are needed to span the full range of Acquisition Strategies that the prototype would be expected to develop?

- Personnel: How many experts will be needed to supply the requisite knowledge? Who are they, and how much time can they devote to the project?

- Special computing resources: Are there any special facilities (e.g., software, access to databases, graphics) that are required for completion of the prototype?

Appendix B

ACQUISITION STRATEGY FOR LAMP-H

1. Program Structure

The LAMP-H Program will be managed by the PM-AWC throughout system acquisition, production, and fielding. The PM-AWC's responsibility will include P3I, supportability, and life cycle cost. Engineering effort in support of the LAMP-H program will be provided by TROSCOM's BRDC for the PM-AWC.

The Program will be structured as a tailored acquisition of a Category C1/C2 Nondevelopmental Item (NDI) - a new system assembled from components available through the commercial market, some of which may have to go additional R&D. Market surveys indicate that the NDI approach will have low risk.

The Program will consist of a competitive Proof of Principle Phase followed by a competition for a single source procurement of 14 LAMP-H craft for the Production and Deployment Phase. Prior to program initiation, a complete set of program management documents will be issued. These include an Operational and Organizational (O&O) Plan, System Concept Paper (SCP), Test and Evaluation Master Plan (TEMP), Market Survey Report, Tentative Basis of Issue Plan (TBOIP), Tentative Qualitative and Quantitative Personnel Requirements Information (TQQPRI), Integrated Logistics Support Plan (ILSP), Procurement Acquisition Plan (PAP), Letter of Agreement (LOA), and a Baseline Cost Estimate (BCE). During the Proof of Principle Phase, competing contractors will be required to bid on the construction of prototype hardware based on a per-

formance oriented specification. This specification will define criteria and goals and give the contractor major responsibility to manage and execute design and manufacture of system hardware and software. Specific tasks to be performed by the contractor will include:

a. Field test his proposed propulsion method(s) on a Government-Furnished Equipment (GFE) hoverbarge to assess and document propulsion and handling trade-offs.

b. Build and test a prototype LAMP-H.

c. Define support system hardware and software.

d. Revise performance specifications and prepare a technical data package (TDP) suitable for competing the production buy.

Upon completion of prototype testing at the end of the Proof of Principle Phase, the Milestone I/III In-Process Review (IPR) will be held. If a negative decision is reached, program cancellation will occur, or the program will be reoriented for further development using a revised acquisition strategy. If a positive Milestone I/III IPR decision is reached, the LAMP-H will be type classified, and a Request for Proposal (RFP) will be released for a competitive production buy of 14 LAMP-H craft using the TDP prepared under sub-paragraph (d) above. A single source procurement will be pursued because of the limited production quantity involved.

A Pre-planned Product Improvement (P3I) Program will be initiated during the Proof of Principle Phase. Planned improvements include increasing the LAMP-H's productivity by decreasing the craft's turnaround time and investigating auxiliary "bolt on" propulsion devices to enhance productivity and expand mission capability.

The NDI strategy for the LAMP-H offers the opportunity to acquire a

state-of-the-art technology hovercraft at reduced cost and reduced time to fielding while still satisfying the user's needs.

2. Contracting Strategy

A cost plus award fee (CPAF) contract is planned for the Proof of Principle Phase with awards on design to unit production cost (DTUPC); Reliability, Availability, and Maintainability (RAM); and schedule performance. A firm fixed price (FFP) contract with warranties will be used for the production phase. Particular attention will be given to acquisition of a TDP suitable for competitive procurement (break out) of parts and components.

3. Tailoring the Acquisition Process

The LAMP-H Acquisition Strategy is tailored as shown by the following list of significant events:

a. Award of Prototype (6.3) Contract to include ILS-JUNE 86

b. GFE Hoverbarge Available to Contractor - JUNE 86-JUNE 87

c. Delivery of Prototype LAMP-H - NOV 87

d. SESAME - NOV 87

e. Prototype Testing - DEC 87 - AUG 88

f. Begin Provisioning - DEC 87

g. Deliver TDP Configuration Freeze - OCT 88

h. Milestone I IPR (NDI Decision) - JAN 89

i. Milestone III IPR (TC Decision) - JAN 89

j. Issue Production Contract IFB - FEB 89

k. Production Contract Award - AUG 89

l. Delivery of Craft #1 and Maintenance Support Package - NOV 90

m. Initial Production Test - DEC 90 - MAR 91

n. Approval of IPT Results and Logistic Support Package - MAR 91

o. SMR Coding Conference - APR 91

p. Acceptance of Craft #1 and #2 - APR 91

q. Completion of Government Provisioning and Logistic Support Package - APR 91

r. First Unit Equipped - MAY 91

s. Delivery of Craft #3 - AUG 91

t. Delivery of Craft #4 - OCT 91

u. Delivery of Craft #5 - JAN 92

v. Delivery of Craft #6 - APR 92

w. Delivery of Final Craft (#14) - APR 94

4. Supportability

The LAMP-H will have full organic support using the Standard Army three level maintenance concept: Unit, Intermediate, and Depot.

Supply support will use the standard Army supply system. Modular replacement, rather than unit repair, will be stressed during the Development and Validation phase of the program. A tailored Logistics Support Analysis/ Logistics Support Analysis Record (LSA/LSAR) program will be employed during the prototype contract. The tailoring will be based on the NDI aspect of the LAMP-H. Since the prototype contract will result in a full disclosure TDP and adequate LSAR, the following key ILS elements status will be at Milestone III:

a. Manuals will have been prepared and evaluated, ready for finalization during the production phase.

b. Provisioning will be at a level requiring only an update as a result of the production contract First Article Testing (FAT).

c. The effectiveness of ILS design influence including MANPRINT will have been evaluated via Prototype Testing.

d. Training requirement including New Equipment Training (NET) will be established.

e. Test, Measurement, and Diagnostic Equipment (TMDE) requirements will be finalized.

At First Unit Equipped (FUE), which will be under Total Package/Unit Material Fielding (TP/UMF), full organic support will be available. To insure this is accomplished, the production contract will stipulate that software deliveries must be accomplished prior to Government acceptance of the end items. The Military Occupational Skill (MOS) for the operator and maintainer will be determined through application of LSA/LSAR (tasks and skill analysis). Maintenance is to be simplified by modular replacement, rather than unit repair, and will be stressed during the prototype phase of the program. The maintenance concept will support the TP/UMF policies and procedures.

The ILSP will address specific plans for implementation to accomplish this strategy.

5. Manufacturing and Production

The integration of engineering and production management will be a continuous process that will commence early in the advanced development phase of the program. The following activities are planned to bring the LAMP-H system to a start of production readiness:

a. A complete, fully documented TDP consisting of drawings and specifications will be prepared from the effort of the prototype contractor. Any design changes that occur between the prototype and production craft will be controlled and processed as Engineering Change Proposals (ECP's). Producibility analysis, together with a review of manufacturing technology development, will be accomplished during the

advanced development phase to ensure that there is a steady rate of progress toward the Production and Deployment Phase.

b. A Value Engineering (VE) effort will be implemented by the prototype and production contractor. VE is necessary because of the relatively high unit price of the LAMP-H. Contractor and government engineers will continually search for ways to reduce cost.

There are a moderate number of companies capable of manufacturing the LAMP-H. The largest and most well known in the industrial base include: Bell Aerospace-Textron, Lockheed Marine, Aerojet Techsystems Company, and RMI. All of these companies have the financial and technological base to successfully fabricate the LAMP-H. The economic production rate has not been established at this point in the program. A unit production cost study by the developer for the LAMP-H was conducted and incorporated into the Baseline Cost Estimate, 30 August 1984.

6. Test and Evaluation

The contractors will be given major responsibilities for designing, planning, and executing the technical testing required for proof of principle and prototype demonstration and qualification. The critical test issues for designing and planning tests are as stated in the pertinent program management documents (Independent Evaluation Plan (IEP), TEMP) and the performance specification. All contractor test activities will be closely monitored by technically qualified Government personnel.

In accordance with AR 70-10 (7 June 1985), Paragraph 6-10.(3)(c), for a new system assembled from components, testing is required in a military environment. Moreover, preproduction testing of the complete system, hardware/ software integration tests, user tests, and first article tests will also be required. The necessary tests, the testers,

the evaluators, and evaluation activities are delineated in the draft TEMP. The principles of Continuous Evaluation (CE) will be practiced throughout the program.

7. Cost Growth and Driver

An important element in controlling cost growth is to maintain program stability. Program schedule changes resulting from Congressional budget cuts can mean cost growth. Changes to the LAMP-H system must be avoided where possible and preferably treated as candidates for product improvements. Pre-planned product improvements will be proposed as funding category 6.2 efforts in FY87 and FY88. Where it is feasible to look ahead to these pre-planned product improvements, the LAMP-H shall be designed to accommodate them. An example of this is the incorporation of tow points on the craft to enable towing/pushing by an enhanced auxiliary amphibian vehicle (Air Cushion Tug Concept).

Acquisition strategy will be used as a tool to establish both Acquisition and Operating and Support (O&S) cost controls. These controls will be exercised during four major program efforts:

a. Project Definition/Preparation of Solicitation

b. Proposal Evaluation

c. Prototype Contract

d. Production Contract

During Project Definition, Design to Unit Production Cost (DTUPC goals will be derived from the BCE and independent cost studies. RAM goals will be developed by Belvoir Research and Development Center prior to issuance of the Request for Proposal (RFP).

These factors have been selected for control because they are the largest contributors to the System Life Cycle cost. Production costs,

along with prototype contract costs, are the major contributors to acquisition cost. RAM is a major determinant of O&S cost since it drives parts replacement, inventory, supply pipeline, and maintenance expenditures. Productivity influences both acquisition and O&S cost since it determines the number of craft that must be procured and operated in order to do the Logistics Over the Shore (LOTS) mission.

During proposal evaluation, a Government cost analysis will be run on prototype contract cost and unit production cost estimates. An evaluation of LAMP-H RAM and performance will also be accomplished and a first cut at O&S cost estimates for the various designs will be made. These analyses will become factors in contractor selection. Contract cost and design-to-cost (DTC) will be included in the CPA' Prototype Contract. Cost/Schedule Status Reports (C/SSR) will be employed to insure contract cost visibility as well as report on actual prototype fabrication costs. This data will be used to establish cost credibility and will be used in negotiating the production contract.

An FFP contract will be used for production to minimize acquisition cost risk. Production risks will be minimized by the use of warranties.

Intensive contract monitoring is another strong element for containing cost growth. The Ship Work Breakdown Structure (SWBS) will be used for planning, organizing, and managing all contract work. The C/SSR reporting by SWBS will be required for government document review. Government Program and Design reviews will be used as another method of controlling costs.

The contractor will be required to implement a Material Deterioration Prevention and Control Plan (MADPAC) as another method of controlling costs.

8. Technical Risks

The Requirements/Technical Base Activities Phase consisted of evaluation of four design concepts:

a. Modification of a British SR.N4 hoverferry

b. Modification of a U.S. Navy LCAC assault craft

c. Modification of a U.S. Army LACV-30 lighter

d. Modification of a commercial hoverbarge design

An evaluation was conducted of all four concepts with a final recommendation that a commercial type self-propelled hovercraft was the best approach for the LAMP-H Program. The technical risks associated with commercial type air cushion vehicle (ACV) systems are considered low, while they offer a very high potential for providing the Army with the capabilities specified in the LAMP-H Required Operational Capability (ROC). Market surveys for the LAMP-H indicate that the state-of-the-art technology is available to meet Army LAMP-H requirements through fabrication of a new system using commercially available components. Technical risks associated with various propulsion methods (wheeled, water propellers) and loading ramps will be reduced to an acceptable level through initial investigations and demonstrations of propulsion systems installed on a government-furnished hoverbarge, and through the use of prototypes.

9. Safety and Health

A system safety and health program has been planned and incorporated into the prototype solicitation to ensure that system safety and health concerns are addressed throughout the design process and that the LAMP-H system is designed for maximum safety consistent with MIL-STD-382A, System Safety Requirements. System safety documenta-

tion for the LAMP-H program includes a System Safety Program Plan and System Safety Hazard Data Sheet.

10. Soldier-Machine Interface

Full MANPRINT concepts will be addressed throughout the prototype/production fabrication and testing of the LAMP-H in accordance with MIL-STD-1472. U.S. Army Troop Support Command (TROSCOM's) Human Engineering Officer has provided standards for the prototype/production contract statement of work and, as a part of the Government team, will monitor contractor performance. In addition, MANPRINT initiatives will be incorporated into the program.

11. Rationalization, Standardization, Interoperability (RSI)

Possible RSI items for the Air Cushion Vehicle System are fuels, lubricants, and diesel engine parts. Standardization and interoperability of the system with those of our NATO allies will be considered throughout the material acquisition process. There are no known requirements for the LAMP-H as a NATO standardized item of equipment. No existing STANAGS/QSTAGS or other agreements addressing the LAMP-H are in existence at this time. At the present time, the Army's LAMP-H will be stationed at Fort Story, VA and not stationed overseas in foreign lands.

12. Survivability and Endurance

NBC Contamination/decontamination protection is required for the system.

The system must be capable of operation in and up to Sea State 3 with full payload (140 tons) for a minimum of 10 hours. Integration/user testing will be required to validate this endurance.

Award fees will be used for incentives during the prototype

contract to assure goals are met for KDTC, RAM, and Scheduling.

13. Short Term Issues

None identified at this time.

Appendix C

PARAGRAPH 1 RULE BASE

In this appendix, more detail is provided about the paragraph 1 rule base. First, a listing of the data base attributes that are used and manipulated within the rule base is shown. The rules that use each data base attribute are listed, and an indication is given to show if the data base attribute is an input to the rule (used in the "if" portion of the rule), an output of the rule (the value of the attribute is changed in the "then" portion of the rule or by some action initiated in the "then" clause), and whether or not the user has any influence on the value given to the attribute.

After the manipulation of the data base attributes within the rule base is listed, an English description of all the rules in the rule base is provided. This is followed by a short description of the "perform" files that are referenced within the rules.

DATABASE ATTRIBUTES USED WITHIN THE PARAGRAPH 1 RULE BASE:

DATABASE ATTRIBUTES	RULES	INPUT	OUTPUT	USER INFLUENCES
NDI Procurement or Not	FINDNDI	*	*	*
	NDIKNOWN	*		
	NOTNDI	*		
NDI Category	FINDNDI	*	*	*
	NDIKNOWN	*		
	SYSNDIA	*		
	SYSNDIB	*		
	SYSNDIC	*		
	SYSNDIAM	*		
	SYSNDIBP	*		
Status of Program Summary	GETSUM	*	*	*
	SUMGOT	*		
	SUMLATER	*		
	SYSNDIA	*		
	SYSNDIB	*		
	SYSNDIC	*		
	SYSNDIAM	*		
	SYSNDIBP	*		
Should Assistant Discuss NDI?	GETSUM			*
	SUMGOT		*	
	SYSNDIA	*		
	SYSNDIB	*		
	SYSNDIC	*		
	SYSNDIAM	*		
	SYSNDIBP	*		
Have Program Responsibility Defaults Been Presented?	DFALTS	*		
Materiel Developer	DFALTS		*	*
	NEEDORGS	*	*	*
	GOTORGS	*		
	MDDRDIFF	*		
	MDDRSAME	*		
Combat Developer	DFALTS		*	*
	NEEDORGS	*	*	*
	GOTORGS	*		
	KNOWCD	*		

DATABASE ATTRIBUTES	RULES	INPUT	OUTPUT	USER INFLUENCES
Proponent School	DFALTS		*	*
	NEEDORGS	*	*	*
	GOTORGS	*		
	KNOWPS	*		
Organization with Development Responsibility	DFALTS		*	*
	NEEDORGS	*	*	*
	GOTORGS	*		
	MDDRDIFF	*		
	MDDRSAME	*		
Organization with Production, Procurement, and Readiness Responsibility	DFALTS		*	*
	NEEDORGS	*	*	*
	GOTORGS	*		
	KNOWPPRR	*		
System Integrator	DFALTS		*	*
	NEEDORGS	*	*	*
	GOTORGS	*		
	KNOWINT	*		
System Logistician	DFALTS		*	*
	NEEDORGS	*	*	*
	GOTORGS	*		
	KNOWLOG	*		
Development Test Evaluator	DFALTS		*	*
	NEEDORGS	*	*	*
	GOTORGS	*		
	DTETDIFF	*		
	DTETSAME	*		
	DTEONLY	*		
	DTTONLY	*		
Development Test Tester	DFALTS		*	*
	NEEDORGS	*	*	*
	GOTORGS	*		
	DTETDIFF	*		
	DTETSAME	*		
	DTEONLY	*		
	DTTONLY	*		

DATABASE ATTRIBUTES	RULES	INPUT	OUTPUT	USER INFLUENCES
Operational Test Evaluator	DFALTS		*	*
	NEEDORGS	*	*	*
	GOTORGS	*		
	OTETDIFF	*		
	OTETSAME	*		
	OTEONLY	*		
	OTTONLY	*		
Operational Test Tester	DFALTS		*	*
	NEEDORGS	*	*	*
	GOTORGS	*		
	OTETDIFF	*		
	OTETSAME	*		
	OTEONLY	*		
	OTTONLY	*		
DESCOM Depot	NEEDORGS	*	*	*
	GOTORGS	*		
	MANTDESC	*		
	MANTCONT	*		
	MANTLTR	*		
	MANTUNIT	*		
To What Extent Will Contractor Support Be Used?	GETSPT	*	*	*
	SPTGOT	*		
	SPTALL	*		
	SPTDEV	*		
	SPTPDN	*		
Will GFE/GFI/GFP Be Provided to the Contractor?	GETGFE	*	*	*
	GFEYES	*		
	GFENO	*		

ENGLISH DESCRIPTION OF PARAGRAPH 1 RULE BASE:

Rule: FINDNDI

IF: It is unknown whether the procurement will be NDI; OR
The NDI category is unknown

THEN: Consult the NDI rule base

Rule: NDIKNOWN

IF: It is known whether the procurement is NDI; AND
The NDI category is known

THEN: Change NDIDONE to true

Rule: NOTNDI

IF: The procurement will not be NDI

THEN: Tell the user no help can be given since the procurement is not NDI;
End the consultation with the Paragraph 1 rule base

Rule: GETSUM

IF: The user has not had the opportunity to enter the program summary yet

THEN: Perform PGMSUM

Rule: SUMGOT

IF: The program summary has been entered

THEN: Add the summary to the paragraph 1 output file;
Add a line to the justification file indicating the user entered the program summary

Rule: SUMLATER

IF: The user deferred entering of the program summary until later

THEN: Add a statement indicating the program summary will be developed later to the paragraph 1 output and justification files

Rule: SYSNDIA

IF: The program summary has been entered; and
The user indicated the assistant should provide a discussion about NDI; and
The NDI category is "A"

THEN: Add text for category "A" NDI discussion to the paragraph 1 output file;
Add a statement to the justification file indicating that the assistant provided the NDI discussion

Rule: SYSNDIB

IF: The program summary has been entered; and
The user indicated the assistant should provide a discussion about NDI; and
The NDI category is "B"

THEN: Add text for category "B" NDI discussion to the paragraph 1 output file;
Add a statement to the justification file indicating that the assistant provided the NDI discussion

Rule: SYSNDIC

IF: The program summary has been entered; and
The user indicated the assistant should provide a discussion about NDI; and
The NDI category is "C"

THEN: Add text for category "C" NDI discussion to the paragraph 1 output file;
Add a statement to the justification file indicating that the assistant provided the NDI discussion

Rule: SYSNDIAM

IF: The program summary has been entered; and
The user indicated the assistant should provide a discussion about NDI; and
The NDI category is "A-"

THEN: Add text for category "A-" NDI discussion to the paragraph 1 output file;
Add a statement to the justification file indicating that the assistant provided the NDI discussion

Rule: SYSNDIBP

IF: The program summary has been entered; and
The user indicated the assistant should provide a discussion about NDI; and
The NDI category is "B+"

THEN: Add text for category "B+" NDI discussion to the paragraph 1 output file;
Add a statement to the justification file indicating that the assistant provided the NDI discussion

Rule: DFALTS

IF: The user has not had the opportunity to select/reject default values for which organizations have various program responsibilities

THEN: perform DEFAULTS

Rule: NEEDORGS

IF: The materiel developer is not known; OR
The combat developer is not known; OR
The proponent school is not known; OR
The organization with development responsibility is not known; OR
The organization with production, procurement, and readiness responsibility is not known; OR
The system integrator is not known; OR
The system logistician is not known; OR
The developmental test evaluator is not known; OR
The developmental test tester is not known; OR
The operational test evaluator is not known; OR
The operational test tester is not known; OR
The DESCOM depot is not known

THEN: perform AGENCIES

Rule: GOTORGS

IF: The materiel developer is known; AND
The combat developer is known; AND
The proponent school is known; AND
The organization with development responsibility is known; AND
The organization with production, procurement, and readiness responsibility is known; AND
The system integrator is known; AND
The system logistician is known; AND

The developmental test evaluator is known; AND
The developmental test tester is known; AND
The operational test evaluator is known; AND
The operational test tester is known; AND
The DESCOM depot is known

THEN: Add a statement to the justification file indicating that the user has picked the responsible agencies

Rule: MDDRDIFF

IF: The materiel developer is known; AND
The materiel developer does not have development responsibility

THEN: Add statement to the paragraph 1 output file identifying the organization that is the materiel developer and the organization that has development responsibility

Rule: MDDRSAME

IF: The materiel developer is known; AND
The materiel developer also has development responsibility

THEN: Add statement to the paragraph 1 output file indicating which organization is the materiel developer and that this same organization has development responsibility for the program

Rule: KNOWCD

IF: The combat developer is known

THEN: Add a statement to the paragraph 1 output file indicating which organization is the combat developer for the program

Rule: KNOWPS

IF: The proponent school is known

THEN: Add a statement to the paragraph 1 output file identifying the proponent school for the program

Rule: KNOWPPRR

IF: The organization with production, procurement, and readiness responsibility is known

THEN: Add a statement to the paragraph 1 output file identifying the organization with PPR responsibility for the program

Rule: KNOWINT

IF: The system integrator is known

THEN: Add a statement to the paragraph 1 output file identifying the system integrator for the program

Rule: KNOWLOG

IF: The system logistician is known

THEN: Add a statement to the paragraph 1 output file identifying the system logistician for the program

Rule: DTEDDIFF

IF: The developmental test evaluator is known; AND
The developmental test tester is known; AND
The DT evaluator is not the DT tester

THEN: Add a statement to the paragraph 1 output file identifying which organization has DT evaluator responsibility and which organization has DT tester responsibility

Rule: DTETSAME

IF: The developmental test evaluator is known; AND
The developmental test tester is known; AND
The DT evaluator is also the DT tester

THEN: Add a statement to the paragraph 1 output file identifying the organization with DT test and evaluation responsibility

Rule: DTEONLY

IF: The developmental test evaluator is known; AND
No developmental test tester has been assigned

THEN: Add a statement to the paragraph 1 output file identifying the organization with DT evaluator responsibility

Rule: DTTONLY

IF: The developmental test tester is known; AND
No developmental test evaluator has been assigned

THEN: Add a statement to the paragraph 1 output file identifying the organization with DT tester responsibility

Rule: OTEDDIFF

IF: The operational test evaluator is known; AND
The operational test tester is known; AND
The OT evaluator is not the OT tester

THEN: Add a statement to the paragraph 1 output file identifying which organization has OT evaluator responsibility and which organization has OT tester responsibility

Rule: OTETSAME

IF: The operational test evaluator is known; AND
The operational test tester is known; AND
The OT evaluator is also the OT tester

THEN: Add a statement to the paragraph 1 output file identifying the organization with OT test and evaluation responsibility

Rule: OTEONLY

IF: The operational test evaluator is known; AND
No operational test tester has been assigned

THEN: Add a statement to the paragraph 1 output file identifying the organization with OT evaluator responsibility

Rule: OTTONLY

IF: The operational test tester is known; AND
No operational test evaluator has been assigned

THEN: Add a statement to the paragraph 1 output file identifying the organization with OT tester responsibility

Rule: MANTDESC

IF: The DESCOM depot has been assigned

THEN: Add a statement to the paragraph 1 output file identifying the DESCOM depot that will be responsible for maintenance of the item

Rule: MANTCONT

IF: Contractor maintenance will be used for the item throughout the life cycle

THEN: Add a statement to the paragraph 1 output file indicating that no DESCOM depot will be assigned because contractor maintenance will be used throughout the life cycle

Rule: MANTLTR

IF: The DESCOM depot is not assigned now but will be assigned later

THEN: Add a statement to the paragraph 1 output file indicating that the DESCOM depot has not yet been assigned

Rule: MANTUNIT

IF: All maintenance on the item will be performed at the unit level

THEN: Add a statement to the paragraph 1 output file indicating that no DESCOM depot will be assigned since all maintenance for the item will be done at unit level

Rule: GETSPT

IF: No knowledge exists regarding the use of the total contractor support approach for the program

THEN: Perform SPTFORM

Rule: SPTGOT

IF: The user entered a discussion reference total contractor support

THEN: Add the text entered by the user to the paragraph 1 output file
Add a statement to the justification file indicating the discussion regarding contractor support was entered by the user

Rule: SPTALL

IF: Total contractor support will be used throughout the life cycle

THEN: Add a statement to the paragraph 1 output file indicating that total contractor support will be used throughout the item life cycle
Add a statement to the justification file indicating that the user said contractor support would be used throughout the life cycle

Rule: SPTDEV

IF: Total contractor support will be used in the development phase only

THEN: Add a statement to the paragraph 1 output file indicating that total contractor support will be used initially, with maintenance transitioning to the normal Army maintenance system
Add a statement to the justification file indicating that the user said contractor support would be used in the development phase only

Rule: SPTPDN

IF: Total contractor support will be used in the production phase only

THEN: Add a statement to the paragraph 1 output file indicating that normal Army maintenance will be used initially, with a transition to total contractor support
Add a statement to the justification file indicating that the user said contractor support would be used in the production phase only

Rule: GETGFE

IF: No knowledge exists regarding Government Furnished Equipment (GFE), Government Furnished Property (GFP), and Government Furnished Items (GFI)

THEN: perform GFE

Rule: GFEYES

IF: The user entered a discussion regarding GFE/GFP/GFI

THEN: Add the text entered by the user to the paragraph 1 output file
Add a statement to the justification file indicating that the discussion was entered by the user

Rule: GFENO

IF: No GFE/GFP/GFI will be provided to the contractor

THEN: Add a statement the the paragraph 1 output file indicating that no GFE/GFP/GFI will be provided to the contractor
Add a statement to the justification file indicating that the user said no GFE/GFP/GFI would be furnished to the contractor

Rule: P1DONE

IF: All required item for paragraph 1 have been addressed

THEN: End the consultation with the paragraph 1 rule base

DESCRIPTION OF "PERFORM" FILES:

PGMSUM : Prompts the user to enter a summary of the program. Allows the user to decide if the assistant should discuss NDI after the program summary, or if the user will include the NDI discussion in the summary, or if paragraph 1 should continue to be developed without entering the program summary at this time.

DEFAULTS: Allows the user to pick one of three "fields of endeavor" in which the item being procured best fits (or pick none of the three). If one of the categories is selected, a list of default organizations is matched against the different program responsibilities. The default organizations are determined based on the category selected. The user can then accept or reject any or all of the defaults. If the user does not select from one of the fields of endeavor, no defaults are presented.

AGENCIES: Prompts the user to enter the organization responsible for all of the progam aspects for which the default organizations were previously rejected (or if no field of endeavor was picked, all program aspects are presented). The user can enter the organization name for each aspect as prompted, or indicate that no organization is assigned that responsibility.

SPTFORM : Prompts the user to indicate if the total contractor support management option was used. The user may or may not be asked to enter text depending on which option is picked from the initial question form.

GFE : Prompts the user to indicate if GFE/GFP/GFI is being furnished to the contractor. If so, the user is prompted to enter text discussing what will be provided and how.

Appendix D

Paragraph 3 Rule Base

In this appendix, more detail is provided about the paragraph 3 rule base. First, a listing of the data base attributes that are used and manipulated within the rule base is shown. The rules that use each data base attribute are listed, and an indication is given to show if the data base attribute is an input to the rule (used in the "if" portion of the rule), an output of the rule (the value of the attribute is changed in the "then" portion of the rule or by some action initiated in the "then" clause), and whether or not the user has any influence on the value given to the attribute.

After the manipulation of the data base attributes within the rule base is listed, an English description of all the rules in the rule base is provided. This is followed by a short description of the "perform" files that are referenced within the rules.

DATABASE ATTRIBUTES USED WITHIN THE PARAGRAPH 3 RULE BASE:

DATABASE ATTRIBUTES	RULES	INPUT	OUTPUT	USER INFLUENCES
NDI Procurement or Not	FINDNDI	*	*	*
	NDIKNOWN	*		
	NOTNDI	*		
NDI Category	FINDNDI	*	*	*
	NDIKNOWN	*		
	CATATXT	*		
	CATBTXT	*		
	CATCTXT	*		
	CATAMTXT	*		
	CATBPTXT	*		
	CATJST	*		
	CATATIME	*		
	CATBTIME	*		
	CATCTIME	*		
	NOEVENTS	*		
	NOEVNTSA	*		
	MISLAST	*		
	MISLASTA	*		
	MICLAST	*		
	MICLASTA	*		
	OOLAST	*		
	ASLAST	*		
	ASLASTA	*		
	ROCLAST	*		
	TCLAST	*		
	TCLASTA	*		
	FUELAST	*		
	FUELASTA	*		
Has User Indicated Which Program Events Have Been Completed?	PREVTIME	*		
	GETTIMES	*	*	
	KNOWTIME	*		
What was Last Program Event Completed?	GETTIMES		*	*
	NOEVENTS	*		
	NOEVNTSA	*		
	MISLAST	*		
	MISLASTA	*		
	MICLAST	*		
	MICLASTA	*		
	OOLAST	*		
	ASLAST	*		
	ASLASTA	*		
	ROCLAST	*		

DATABASE ATTRIBUTES	RULES	INPUT	OUTPUT	USER INFLUENCES
	TCLAST	*		
	TCLASTA	*		
	FUELAST	*		
	FUELASTA	*		
Month Market Investigation Started	KNOWTIME		*	*
Year Market Investigation Started	KNOWTIME		*	*
Month Market Investigation Completed	KNOWTIME		*	*
Year Market Investigation Completed	KNOWTIME		*	*
Month O & O Plan Approved	KNOWTIME		*	*
Year O & O Plan Approved	KNOWTIME		*	*
Month Acquisition Strategy Approved	KNOWTIME		*	*
Year Acquisition Strategy Approved	KNOWTIME		*	*
Month ROC Approved	KNOWTIME		*	*
Year ROC Approved	KNOWTIME		*	*
Month Type Classification Occurred	KNOWTIME		*	*
Year Type Classification Occurred	KNOWTIME		*	*
Month First Unit Equipped	KNOWTIME		*	*
Year First Unit Equipped	KNOWTIME		*	*
Are There Any P3I Plans for Item?	GETP3I	*	*	*
	KNOWP3I	*		
Are Any Waivers Necessary for Item?	GETWAVE	*	*	*
	KNOWWAVE	*		

ENGLISH DESCRIPTION OF PARAGRAPH 3 RULE BASE:

Rule: FINDNDI

IF: It is unknown whether thprocurement will be NDI; OR
The NDI category is unknown

THEN: Consult the NDI rule base

Rule: NDIKNOWN

IF: It is known whether the procurement is NDI; AND
The NDI category is known

THEN: Change NDIDONE to true

Rule: NOTNDI

IF: The procurement will not be NDI

THEN: Tell the user no help can be given since the
procurement is not NDI;
End the consultation with the Paragraph 1 rule base

Rule: CATATXT

IF: The NDI category is "A"

THEN: Add the discussion about tailoring the acquisition
strategy that is appropriate for an NDI category A
procurement to the paragraph 3 output file.

Rule: CATBTXT

IF: The NDI category is "B"

THEN: Add the discussion about tailoring the acquisition
strategy that is appropriate for an NDI category B
procurement to the paragraph 3 output file.

Rule: CATCTXT

IF: The NDI category is "C1"; OR
The NDI category is "C2"

THEN: Add the discussion about tailoring the acquisition

strategy that is appropriate for an NDI category C procurement to the paragraph 3 output file.

Rule: CATAMTXT

IF: The NDI category is "A-"

THEN: Add the discussion about tailoring the acquisition strategy that is appropriate for an NDI category A- procurement to the paragraph 3 output file.

Rule: CATBPTXT

IF: The NDI category is "B+"

THEN: Add the discussion about tailoring the acquisition strategy that is appropriate for an NDI category B+ procurement to the paragraph 3 output file.

Rule: CATJST

IF: The NDI category is known

THEN: Add a statement to the justification file indicating what the NDI category is, the reason, and the source.

Rule: CATATIME

IF: The NDI category is "A"; OR
The NDI category is "A-"

THEN: Set the times used to estimate program event completion dates to those appropriate for this category.

Rule: CATBTIME

IF: The NDI category is "B"; OR
The NDI category is "B+"

THEN: Set the times used to estimate program event completion dates to those appropriate for this category.

Rule: CATCTIME

IF: The NDI category is "C1"; OR
The NDI category is "C2"

THEN: Set the times used to estimate program event completion dates to those appropriate for this category.

Rule: PREVTIME

IF: The user has already provided the completion dates for the program events that have been completed

THEN: Perform LOADTIME

Rule: GETTIMES

IF: The user has not been asked which program events have been completed (and when)

THEN: Perform ASKEVENT

Rule: KNOWTIME

IF: The completed program event times are available

THEN: Perform CALCTIME

Rule: NOEVENTS

IF: No program events have been completed; AND
(The NDI category is not "A"; OR
The NDI category is not "A-")

THEN: Add text to the paragraph 3 output file indicating the estimated completion times for all program events.

Rule: NOEVNTSA

IF: No program events have been completed; AND
(The NDI category is "A"; OR
The NDI category is "A-")

THEN: Add text to the paragraph 3 output file indicating the estimated completion times for all program events appropriate for a category A or A- procurement

Rule: MISLAST

IF: The start of the market investigation was the last program event completed; AND
(The NDI category is not "A"; OR
The NDI category is not "A-")

THEN: Add text to the paragraph 3 output file indicating when the market investigation was started, and estimating when the other program events will be completed

Rule: MISLASTA

IF: The start of the market investigation was the last program event completed; AND
(The NDI category is "A"; OR
The NDI category is "A-")

THEN: Add text to the paragraph 3 output file indicating when the market investigation was started, and estimating when the other program events (appropriate for category A or A- procurements) will be completed

Rule: MICLAST

IF: The completion of the market investigation was the last program event completed; AND
(The NDI category is not "A"; OR
The NDI category is not "A-")

THEN: Add text to the paragraph 3 output file indicating when the market investigation was started, when the market investigation was completed, and estimating when the other program events will be completed

Rule: MICLASTA

IF: The completion of the market investigation was the last program event completed; AND
(The NDI category is "A"; OR
The NDI category is "A-")

THEN: Add text to the paragraph 3 output file indicating when the market investigation was started, when the market investigation was completed, and estimating when the other program events (applicable to category A and A- procurements) will be completed

Rule: OOLAST

IF: The appr val of the O&O plan was the last program event completed; AND
(The NDI category is not "A"; OR
The NDI category is not "A-")

THEN: Add text to the paragraph 3 output file indicating when the market investigation was started, when the market investigation was completed, and when the O&O plan was approved; and estimating when the other program events will be completed

Rule: ASLAST

IF: The approval of the acquisition strategy was the last program event completed; AND
(The NDI category is not "A"; OR
The NDI category is not "A-")

THEN: Add text to the paragraph 3 output file indicating when the market investigation was started, when the market investigation was completed, when the O&O plan was approved and when the acquisition strategy was approved; and estimating when the other program events will be completed

Rule: ASLASTA

IF: The approval of the acquisition strategy was the last program event completed; AND
(The NDI category is "A"; OR
The NDI category is "A-")

THEN: Add text to the paragraph 3 output file indicating when the market investigation was started, when the market investigation was completed, and when the acquisition stratey was approved; and estimating when the other program events (applicable to a category A or A- procurement) will be completed

Rule: ROCLAST

IF: The approval of the ROC was the last program event completed; AND
(The NDI category is not "A";
The NDI category is not "A-")

THEN: Add text to the paragraph 3 output file indicating

when the market investigation was started; when the market investigation was completed; when the O&O plan, the acquisition strategy, and the ROC were approved; and estimating when the other program events will be completed

Rule: TCLAST

IF: Type classification was the last program event completed; AND

(The NDI category is not "A";
The NDI category is not "A-")

THEN: Add text to the paragraph 3 output file indicating when the market investigation was started; when the market investigation was completed; when the O&O plan, the acquisition strategy, and the ROC were approved; when type classification occurred; and estimating when First Unit Equipped will happen

Rule: TCLASTA

IF: Type classification was the last program event completed; AND
(The NDI category is "A"; OR
The NDI category is "A-")

THEN: Add text to the paragraph 3 output file indicating when the market investigation was started, when the market investigation was completed, when the acquisition stratey was approved, and when type classification occurred; and estimating when the First Unit Equipped date will be

Rule: FUELAST

IF: First Unit Equipped was the last program event completed; AND
(The NDI category is not "A";
The NDI category is not "A-")

THEN: Add text to the paragraph 3 output file indicating when all the program events were completed

Rule: FUELASTA

IF: First Unit Equipped was the last program event completed; AND
(The NDI category is "A"; OR
The NDI category is "A-")

THEN: Add text to the paragraph 3 output file indicating when all the program events were completed

Rule: GETP3I

IF: The user has not been asked about Pre-Planned Product Improvement plan

THEN: Perform P3IFORM;
Add the appropriate text to the paragraph 3 output file

Rule: GETWAVE

IF: The user has not been asked about waivers that may be required

THEN: Perform WAVEFORM;
Add appropriate text to the paragraph 3 output file

Rule: KNOWP3I

IF: The user has already been asked about P3I plans

THEN: Add appropriate text to the paragraph 3 output file;
Add a statement to the justification file indicating the user previously addressed P3I

Rule: KNOWWAVE

IF: The user has already been asked about waivers

THEN: Add appropriate text to the paragraph 3 output file;
Add a statement to the justification file indicating that the user previously addressed waivers

Rule: P3DONE

IF: All required aspects of paragraph 3 have been addressed

THEN: End the consultation with the paragraph 3 rule base

DESCRIPTION OF "PERFORM" FILES

LOADTIME: Loads the previously input information regarding completed program events from the data base.

CALCTIME: Computes the estimated completion dates for program events that have not yet been completed. Times are computed based on the NDI category that applies to the program.

ASKEVENT: Prompts the user for information about completed program events, and when they were completed.

P3IFORM : Prompts the user for information about Pre-Planned Product Improvement plans for the item. The user may or may not have to enter a discussion about P3I depending on the choices made.

WAVEFORM: Prompts the user for information about possible waivers for testing requirements, for the total unit fielding concept, for competitive procurement, or any other waivers. The user may or may not have to enter a discussion about one or more of these depending on the choices made.

Appendix E

NDI Rule Base

In this appendix, more detail is provided about the NDI rule base. First, a listing of the data base attributes that are used and manipulated within the rule base is shown. The rules that use each data base attribute are listed, and an indication is given to show if the data base attribute is an input to the rule (used in the "if" portion of the rule), an output of the rule (the value of the attribute is changed in the "then" portion of the rule or by some action initiated in the "then" clause), and whether or not the user has any influence on the value given to the attribute.

After the manipulation of the data base attributes within the rule base is listed, an English description of all the rules in the rule base is provided. This is followed by a short description of the "perform" files that are referenced within the rules.

DATABASE ATTRIBUTES USED WITHIN THE NDI RULE BASE:

DATABASE ATTRIBUTES	RULES	INPUT	OUTPUT	USER INFLUENCES
NDI Procurement or Not	NDIORNOT	*	*	*
	NOTNDI	*		
	FINDCAT	*		
	DECIDENO		*	
NDI Category	FINDCAT	*	*	*
	CATKNOWN	*		
	NEEDHELP	*		
	DECIDEA	*	*	
	DECIDEBP	*	*	
	DECIDEAM	*	*	
	DECIDEB	*	*	
	DECIDEC1	*	*	
	DECIDEC2	*	*	
	DECIDENO	*		
Is Item Available in Commercial Marketplace?	NEEDHELP	*	*	*
	DECIDEA	*		
	DECIDEBP	*		
	DECIDEAM	*		
	DECIDEB	*		
	DECIDEC1	*		
	DECIDEC2	*		
	DECIDENO	*		
Is This Item Currently in Army Inventory?	NEEDHELP	*	*	*
	DECIDEA	*		
	DEDICEBP	*		
	DECIDEAM	*		
	DECIDEB	*		
Will Item Be Used in Same Environment as Used Commercially?	NEEDHELP	*	*	*
	DECIDEA	*		
	DECIDEBP	*		
	DECIDEAM	*		
	DECIDEB	*		
Are Components of the Item Available Commercially?	NEEDHELP	*	*	*
	DECIDEC1	*		
	DECIDEC2	*		
Are All, or Just Some, of the Components Available?	NEEDHELP	*	*	*
	DECIDEC1	*		
	DECIDEC2	*		
	DECIDENO	*		

ENGLISH DESCRIPTION OF NDI RULE BASE:

Rule: NDIORNOT

IF: It is unknown whether the procurement will be NDI

THEN: Perform ASKNDI

Rule: NOTNDI

IF: The procurement will not be NDI

THEN: Tell the user no help can be given since the procurement is not NDI;
End the consultation with the NDI rule base

Rule: FINDCAT

IF: The NDI category is not known; AND
The procurement will be NDI

THEN: Perform ASKCAT

Rule: CATKNOWN

IF: The NDI category is known

THEN: End the consultation with the NDI rule base

Rule: NEEDHELP

IF: The user ask for help deciding the NDI category; AND
It is not known if the item is currently in the Army inventory; AND
It is not known if the item is available in the commercial marketplace; AND
It is not known if the item will be used in the same environment as commercially; AND
It is not known if components of the item are available in the commercial marketplace; AND
It is not known if all or just some of the components are available

THEN: Perform HELPCAT

Rule: DECIDEA

IF: The user ask for help deciding the NDI category; AND
The item is available in the commercial market; AND
The item is currently in the Army inventory; AND
The item will be used in the same environment as it is used commercially

THEN: The NDI category is "A";
End the consultation with the NDI rule base

Rule: DECIDEBP

IF: The user ask for help deciding the NDI category; AND
The item is available in the commercial market; AND
The item is currently in the Army inventory; AND
The item will NOT be used in the same environment as it is used commercially

THEN: The NDI category is "B+";
End the consultation with the NDI rule base

Rule: DECIDEAM

IF: The user ask for help deciding the NDI category; AND
The item is available in the commercial market; AND
The item is NOT currently in the Army inventory; AND
The item will be used in the same environment as it is used commercially

THEN: The NDI category is "A-";
End the consultation with the NDI rule base

Rule: DECIDEB

IF: The user ask for help deciding the NDI category; AND
The item is available in the commercial market; AND
The item is NOT currently in the Army inventory; AND
The item will NOT be used in the same environment as it is used commercially

THEN: The NDI category is "B";
End the consultation with the NDI rule base

Rule: DECIDEC1

IF: The user ask for help deciding the NDI category; AND
The item is NOT available in the commercial market;
AND
Components of the item are available in the
commercial market; AND
All components of the item are available commercially

THEN: The NDI category is "C1";
End the consultation with the NDI rule base

Rule: DECIDEC2

IF: The user ask for help deciding the NDI category; AND
The item is NOT available in the commercial market;
AND
Components of the item are available in the
commercial market; AND
Only some components of the item are availabl
commercially

THEN: The NDI category is "C2";
End the consultation with the NDI rule base

Rule: DECIDENO

IF: The user ask for help deciding the NDI category; AND
The item is NOT available in the commercial market;
AND
Components of the item are NOT available in the
commercial market

THEN: The procurement is not NDI;
Tell the user no help can be given because, since not
even some of the components are commercially
available, the NDI approach cannot be followed;
End the consultation with the NDI rule base

DESCRIPTION OF "PERFORM" FILES:

ASKNDI : Prompts the user to answer if the NDI strategy is appropriate for the program under consideration. If the user indicates information is needed about NDI, ASNDI is performed to provide additional information to the user before a decision has to be made.

ASKCAT : Prompts the user to indicate which NDI category is appropriate for the program. A more detailed breakdown of the NDI categories is provided to the user to assist in the decision. The user can answer with a category, or indicate help is needed in deciding the category.

HELPCAT : Decides the NDI category appropriate to the program, or if the program is actually not NDI at all, based on user answers to a series of questions about the characteristics of the program.

Appendix F

Sample Prototype System Output

This appendix contains a sample of paragraphs 1 and 3 that are generated by the prototype system developed in this research. Also included are the justifications that are generated as a result of the paragraph development process.

The paragraphs 1 and 3 shown in this appendix were developed by Mr. Dan Causey as a part of the prototype system evaluation. The procurement program used as a basis for the paragraphs was the LAMP-H program. The actual acquisition strategy developed for this program was used during the problem assessment phase of this research to become familiar with what an acquisition strategy contains. The "human generated" acquisition strategy for the LAMP-H program is contained at Appendix B.

PARAGRAPH 1 FOR LAMP-H PROGRAM

(GENERATED BY THE PROTOTYPE SYSTEM)

1. PROGRAM STRUCTURE:

The LAMP-H program will take advantage of the fact that similar craft have been produced in the past; even though not for use in the same extreme environments. This will help to expedite fielding of the system. The contracts for the system will be based on performance specifications. This will put hardware and software responsibility in the prototype contractor's control, given the specified design goals.

The program will be structured as a tailored acquisition of a Category C1/C2 non-developmental item (NDI) - a new system assembled from components available in the commercial marketplace (some of which may have to undergo additional R&D). Thus, the traditional acquisition cycle will be shortened to take advantage of design, development, testing, and production efforts already completed in the commercial marketplace. This will allow for reduced time to fielding, reduced system cost, and use of the most current and proven technology.

The Belvoir Research & Development Center (Belvoir) has been assigned the role of materiel developer for the program with the Program Manager, Amphibious Water Craft (PM-AWC) having development responsibility. The Training and Doctrine Command (TRADOC) is the combat developer. The proponent school for the program is the US Army Quartermaster School. The Troop Support Command (TROSCOM) has procurement, production, and readiness responsibility. The PM-AWC has been assigned as the system integrator. The logistician for the program is the Logistics Evaluation

Agency (LEA). For the system Development Test (DT), the Test and Evaluation Command (TECOM) is the evaluator with the Aberdeen Proving Ground (APG) having testor responsibility. The US Army Quartermaster School will have test and evaluation responsibility for the system Operational Testing (OT). No DESCOM Depot is required since all maintenance will be performed at the unit level.

The total contractor support management option will be used throughout the life cycle. No Government Furnished Equipment (GFE), Items (GFI) or Property (GFP) will be provided.

PARAGRAPH 1 JUSTIFICATIONS FOR LAMP-H PROGRAM

(GENERATED BY THE PROTOTYPE SYSTEM)

JUSTIFICATIONS FOR PARAGRAPH 1:

The program summary information was input from the user. Discussion reference NDI came from the system (user picked this option). The organizations assigned various responsibilities are a mixture of system defaults accepted by the user and names input by the user. The user indicated that the total contractor support management option would be used in all phases. The user indicated that no GFE/GFP/GFI would be provided.

PARAGRAPH 3 FOR LAMP-H PROGRAM

(GENERATED BY THE PROTOTYPE SYSTEM)

3. TAILORING THE ACQUISITION PROCESS:

The normal acquisition cycle will be compressed for this program. Since this is an NDI category C1/C2 procurement, actions taken to accelerate the process include the combination of the Demonstration and Validation phase with the Full Scale Development phase into a single Production Prove-Out phase. Testing to be conducted will include feasibility/suitability tests, pre-production tests, integration tests, and user tests.

Major efforts in the Proof of Principal phase include the market investigation, concept evaluation testing, and preparation of the acquisition strategy. Upon approval of the NDI acquisition strategy at the Milestone I/II IPR, the Production Prove-Out phase will begin. Significant efforts in this phase will include pre-production quality testing, integration testing, and production specification development. Upon successful completion of the Milestone III IPR for generic type classification of the system, the Production and Deployment phase will begin. Major efforts in this phase will consist of type classification standard, contract award, First Article Test/ Follow-On Evaluation, new equipment training, and operational deployment.

The market investigation was started in MAR 85 and was completed in DEC 85. The O & O Plan was approved in JUL 85, the Acquisition Strategy in JUN 86, and the ROC in DEC 86. Type Classification is expected by DEC 87 with an estimated FUE date of MAY 89.

The LAMP-H system will take advantage of P3I concepts to be fielded as a system using the state-of-the-art propulsion and then be upgraded as the technology progresses in the area of diesel engine powered, multi-dimensional propulsors. The P3I will not expand the system beyond the range defined in the requirements documents; however, it will more fully meet all aspects of the Army's needs. No waivers or exceptions to policy are anticipated for the system.

PARAGRAPH 3 JUSTIFICATIONS FOR LAMP-H PROGRAM

(GENERATED BY THE PROTOTYPE SYSTEM)

JUSTIFICATIONS FOR PARAGRAPH 3:

The NDI category for the program is 'C2' because the item is not commercially available but SOME components are (per the user). The market investigation start and end dates, and O&O, AS, and ROC approval dates were input by the user; all other dates were calculated by the Assistant. Information about P3I plans was entered by the user. The user stated that no waivers or exceptions to policy were planned for the system.

Bibliography

1. Air Force Systems Command. CGADS User's Manual. Hanscom, AFB: Directorate of Executive Information Systems, July 1985.

2. Army Materiel Command/Training and Doctrine Command. Materiel Acquisition Handbook. AMC Pamphlet 70-2/TRADOC Pamphlet 70-2. Headquarters, AMC/TRADOC, 1987.

3. Barr, Avron and Feigenbaum, Edward A. The Handbook of Artificial Intelligence, Volume 1. Stanford: Heuristech Press, 1981.

4. Berry, Dianne C. and Broadbent, Donald E. "Expert Systems and the Man-Machine Interface," Expert Systems - The International Journal of Knowledge Engineering, 3 (4): 228-231 (October 1986).

5. Bobrow, Daniel G. and others. "Expert Systems: Perils and Promise," Communications of the ACM, 29 (9): 880-894 (September 1986).

6. Boose, John H. "Rapid Acquisition and Combination of Knowledge From Multiple Experts in the Same Domain," IEEE Second Conference on Artificial Intelligence Applications, 461-466, (December 1986).

7. Boose, John H. and Jeffrey M. Bradshaw. "NeoETS: Capturing Expert Systems Knowledge in Hierarchical Rating Grids," 1986 Expert Systems in Government Symposium, 34-45, (October 1986).

8. Brownston, Lee and others. Progamming Expert Systems in OPS5. Reading MA: Addison-Wesley Publishing Company, 1985.

9. Causey, Dan. "Evaluation of Expert System for Acquisition Strategies," Memorandum For Record, US Army Belvoir Research, Development, and Engineering Center (STRBE-HL), 26 October 1987.

10. Citrenbaum, Ronald and others. "Selecting a Shell," AI Expert, 30-39, (September 1987).

11. Conklin, Jeffrey. "Hypertext: An Introduction and Survey," Computer, 17-41, (September 1987).

12. Defense Systems Management College. The Program Manager's Support System. PMSS Directorate, Department of Research and Information, 1 January 1987.

13. Hall, Lawrence O. and Wyllis Bandler. "Relational Knowledge Acquisition," IEEE Second Conference on Artificial Intelligence Applications, 509-513, (December 1985).

14. Harmon, Paul and David King. Expert Systems: Artificial Intelligence in Business. New York: John Wiley & Sons, Inc., 1985.

15. Hoffman, Robert R. "The Problem of Extracting the Knowledge of Experts from the Perspective of Experimental Psychology," AI Magazine, 53-67, (summer 1987).

16. Kitto, Catherine M. and John H. Boose. "Heuristics for Expertise Transfer: The Automatic Management of Complex Knowledge Acquisition Dialogs," 1986 Expert Systems in Government Symposium, 53-63, (October 1986).

17. LeClair, Steven R., A Multiexpert Knowledge System Architecture for Manufacturing Decision Analysis. PhD Dissertation, Arizona State, (May 1985).

18. Level Five Research, Inc. Insight 2+ Reference Manual. Indialantic, FL: 1986.

19. Margot, Bruce. "Testing Your Knowledge Base," AI Expert, 42-47, (August 1987).

20. McDermott, John. "R1: The Formative Years," AI Magazine, 21-29, (summer 1981).

21. McGraw, Karen L. and Mary Seale. "Knowledge Elicitation With Multiple Experts: Considerations and Techniques," Artificial Intelligence Review, Second Edition (to be published).

22. Micro Data Base Systems, Inc. Guru Reference Manual,Volumes 1 and 2. Lafayette, IN: October 1986.

23. Mittal, Sanjay and Clive L. Dym. "Knowledge Acquisition from Multiple Experts," AI Magazine, 32-36, (Summer 1985).

24. National Aeronautics and Space Administration. Expert Systems Handbook. 86-FM-19. Houston: Lyndon B. Johnson Space Center, July 1986.

25. Nguyen, Tin A. and others. "Knowledge Base Verification," AI Magazine, 69-75, (Summer 1987).

26. Prerau, David S. "Selection of an Appropriate Domain for an Expert System," AI Magazine, 26-30, (summer 1985).

27. Prerau, David S. "Knowledge Acquisition in the Development of a Large Expert System," AI Magazine, 43-51, (summer 1987).

28. Production Systems Technologies, Inc. OPS83 User's Manual. July 1986.

29. Rolandi, Walter G. "Knowledge Engineering in Practice," AI Expert, 58-62, (December 1986).

30. Rosenberg, Jerry M. Dictionary of Artificial Intelligence and Robotics. New York: John Wiley & Sons, Inc., 1986.

31. Tanimoto, Steven L. The Elements of Artificial Intelligence. Rockville, MD: Computer Science Press, 1987.

32. Texas Instruments, Inc. Personal Consultant Plus User's Guide. Dallas: 1986.

33. The Analytic Sciences Corporation. Acquisition Expert System (AES) Tutorial. Washington, DC: Navy Materiel Command, April 1985.

34. Walters, Richard C. "KBEmacs: Where's the AI?," AI Magazine, 47-56, (Spring 1986).

35. Waterman, Donald A. A Guide to Expert Systems. Reading MA: Addison-Wesley Publishing Company, 1986.

VITA

Captain Robert J. Hammell II was born on 23 August 1957 in San Antonio, Texas. He attended Trumann High School in Trumann, Arkansas, where he graduated with honors in 1975. After high school, CPT Hammell attended Arkansas State University and graduated with High Scholastic Honors in 1979 with a Bachelor of Science degree in Business Data Processing.

Following graduation, CPT Hammell received a Regular Army commission through the ROTC program. He spent four years in a tactical Signal battalion at Fort Bragg, North Carolina. He then attended the Signal Officer's Advanced Course at Fort Gordon, Georgia and the Teleprocessing Operations Officer Course II at the Air Force Institute of Technology (AFIT) at Wright Patterson AFB, Ohio. In April 1984, CPT Hammell was assigned to the Tactical Management Information Systems (TACMIS) Project Management Office. For the last sixteen months of his assignment at TACMIS, CPT Hammell was the Deputy Product Manager for the Tactical Army Combat Service Support Computer System (TACCS).

In September 1986, CPT Hammell entered the AFIT School of Engineering. He has been selected to remain at AFIT as an instructor in the Department of Electrical and Computer Engineering upon graduation.

Military Address: Air Force Institute of Technology
AFIT/ENG
Wright Patterson AFB, OH 45433

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1a. REPORT SECURITY CLASSIFICATION UNCLASSIFIED	1b. RESTRICTIVE MARKINGS
2a. SECURITY CLASSIFICATION AUTHORITY	3. DISTRIBUTION/AVAILABILITY OF REPORT Approved for public release; distribution unlimited
2b. DECLASSIFICATION/DOWNGRADING SCHEDULE	
4. PERFORMING ORGANIZATION REPORT NUMBER(S) AFIT/GCS/ENG/88M-2	5. MONITORING ORGANIZATION REPORT NUMBER(S)
6a. NAME OF PERFORMING ORGANIZATION School of Engineering 6b. OFFICE SYMBOL (If applicable) AFIT/ENG	7a. NAME OF MONITORING ORGANIZATION
6c. ADDRESS (City, State, and ZIP Code) Air Force Institute of Technology Wright-Patterson AFB, Ohio 45433	7b. ADDRESS (City, State, and ZIP Code)
8a. NAME OF FUNDING/SPONSORING ORGANIZATION Defense Systems Mgmt College 8b. OFFICE SYMBOL (If applicable) DRI-S	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER
8c. ADDRESS (City, State, and ZIP Code) Director, PMSS Directorate Fort Belvoir, Virginia 22060-5426	10. SOURCE OF FUNDING NUMBERS PROGRAM ELEMENT NO / PROJECT NO / TASK NO / WORK UNIT ACCESSION NO

11. TITLE (Include Security Classification)

A PROTOTYPE KNOWLEDGE-BASED SYSTEM FOR DEVELOPING ACQUISITION STRATEGIES

12. PERSONAL AUTHOR(S)

[illegible]bert J. Hammell II, B.S., CPT, USA

13a. TYPE OF REPORT	13b. TIME COVERED	14. DATE OF REPORT (Year, Month, Day)	15. PAGE COUNT
MS Thesis	FROM _____ TO _____	1987 December	167

16. SUPPLEMENTARY NOTATION

17. COSATI CODES			18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
FIELD	GROUP	SUB-GROUP	Procurement, Acquisition, Artificial Intelligence, Expert Systems, Acquisition Strategy, Knowledge-Based System, Knowledge Engineering, Knowledge Acquisition, Program Mgmt
12	09		
15	05		

19. ABSTRACT (Continue on reverse if necessary and identify by block number)

Thesis Chairman: Stephen E. Cross, Major, USAF
Adjunct Professor of Computer and Electrical Engineering

The Acquisition Strategy is one of the most important program management documents. The diverse nature of the information required in an acquisition strategy, coupled with frequent changes in acquisition policy and normal loss of organizational expertise in a military environment, create significant problems for the developer of an acquisition strategy.

This thesis demonstrates the effectiveness of applying knowledge-based system techniques to the problem of developing acquisition strategies for government procurements. The need for automated assistance for generating an acquisition strategy is examined, and the reasons why conventional programmin techniques are inadequate for this problem are discussed. The (cont'd)

20. DISTRIBUTION/AVAILABILITY OF ABSTRACT ☒ UNCLASSIFIED/UNLIMITED ☐ SAME AS RPT ☐ DTIC USERS	21. ABSTRACT SECURITY CLASSIFICATION UNCLASSIFIED
22a. NAME OF RESPONSIBLE INDIVIDUAL Stephen E. Cross, Major, USAF	22b. TELEPHONE (Include Area Code) 513-255-5800 / 22c. OFFICE SYMBOL AIAO

DD Form 1473, JUN 86 *Previous editions are obsolete*

BOX 19, ABSTRACT (contd)

development of a prototype knowledge-based system using Guru is outlined, and an evaluation of the system is presented.

The prototype system addresses Non-Developmental Item (NDI) procurements, and two of the thirteen required acquisition strategy paragraphs are implemented. A capability to edit the paragraphs is provided, and consistency between the paragraphs is forced during paragraph development as well as during paragraph editing.

This thesis concluded that a knowledge-based "intelligent assistant" could provide significant benefit to developers of an acquisition strategy. Possible enhancements to extend the prototype and potential areas of further research are also discussed.

END
DATE
FILMED
4-88
DTIC